

Natural Language Knowledge Acquisition Systems

54-61
333524
P-118

N91-70702

End of Year Report

Main PI: Fernando Gomez
Dept. of Computer Science
University of Central Florida
Orlando, FL 32816
Jan-25-91

Table of Contents

Theoretical Components of the Knowledge Acquisition Interface	1
---	---

Appendix A: Viewgraph Description of Research and Interface

Appendix B: Reprint of the Paper: "Knowledge Acquisition from Natural Language for Expert Systems Based on Classification Problem-Solving Methods." *Proceedings of the 4th AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Appendix C: Reprint of the Paper "Finding and Learning Explanatory Connections from Scientific Texts. *Proceedings of the First IEEE Computer Society International Workshop on Tools for Artificial Intelligence*, Fairfax, Virginia.

Appendix D: Reprint of the Paper "Knowledge Acquisition from Natural Language for Expert Systems Based on Classification Problem-Solving Methods." *Journal of Knowledge Acquisition*.

Appendix E: Copy of the Paper "Classification-Based Reasoning" which will appear in the March issue of the journal *IEEE Transactions on Systems, Man and Cybernetics*.

Introduction

The purpose of our task is to automate the acquisition of knowledge from human experts. Our research has proceeded in two ways: (a) advancing the state of the art in knowledge acquisition using natural language and (b) transferring this technology to the actual construction of knowledge acquisition interfaces for real expert systems. As part of our transfer of technology, we are already building a prototype knowledge acquisition interface for OPERA. The interface written in Common Lisp is running in a Symbolics 3653, where we are making use of all graphic features of the Symbolics environment. This report centers in describing the main theoretical components of the knowledge acquisition interface to OPERA, its main milestones and the current effort. In the appendix, the reader may find copies of the papers which have been published during last year on the theoretical aspects of our research.

The goal of OPERA (Expert System Analyst) is to improve the operations support of the computer network in the space shuttle launch processing system. The checkout, control and monitor subsystem (CCMS) is a distributed computer network, which integrates software, microcode, display switches and hardware interface devices. OPERA is intended to function as a consultant to the operations staff assigned to each CCMS task. Two basic expert systems form OPERA: the Real Time System Error Manager (RTSEM) and the Problem Impact Analyst (PIA). When an error occurs, RTSEM displays information on this error obtained from a data base of errors. This information, although based on the CCMS message catalog information, contains experiential knowledge which "resides in the head of the human experts not in texts." The knowledge acquisition bottleneck that the designers of OPERA are presently experiencing is to gather this knowledge from the human experts and transfer it to OPERA in a form assimilable by the data structures and algorithms of the expert system. OPERA contains about one hundred thirty of these errors, but the actual number of errors in the computer network is beyond one thousand. Hence, OPERA is short in its knowledge base by a factor of ten.

The goal of our project is to build a knowledge acquisition interface by means of which a domain expert without knowledge of OPERA or expert systems will be able to transfer his/her

knowledge about the computer network errors to OPERA. One of the goals of this project is to take the English input as entered by an expert and transform it into the data structure which OPERA understands. But there are other goals in our project like helping the expert to organize his/her knowledge, refreshing his/her memory by bringing into place relevant pieces of information that he/she may have forgotten, and analyzing the precision and ambiguity of the English used by him/her to describe errors. (This is a task that we will start this coming year.)

Organization of Knowledge in an Expert Hierarchy

The first task which we have accomplished is to provide a way for the domain expert to organize his/her experiential knowledge. There has been a lot of evidence accumulated showing that human experts organize their knowledge into a hierarchy of concepts. This fact became evident to us during our initial interactions with OPERA experts. Experts use hierarchies all the time to describe the errors of the network system. They use these hierarchies all the time when they analyze errors and determine their causes and corrective steps. But for experts in this domain, the hierarchy is as natural as a botanical hierarchy is for a botanist. The hierarchy is the expert's underlying frame of reference for his/her domain knowledge. *They do not have to learn it.* We have built an initial hierarchy into our knowledge acquisition interface. This hierarchy contains the experiential and heuristic knowledge of the expert, and changes from expert to expert, making true the phrase that each expert has his/her own book. Accordingly, the interface allows different experts to build their own hierarchy. This hierarchy contains the following information:

Knowledge specific to each error message, including:

- * operational advisories
- * diagnostic advisories
- * probable causes

Given this framework, the interaction between the expert and the interface proceeds as follows:

- (1) The expert is asked to select an error from a pre-stored table of errors.
- (2) Then, the expert is asked to classify the error in the hierarchy.

(3) If the expert decides that the hierarchy is not adequately organized, he/she may decide to restructure it to meet his/her criteria. The Interface keeps multiple hierarchies associated with different experts.

(4) Once the expert has classified the error in the hierarchy, he/she is asked to enter information about that error, as explained in the section below.

All this interaction takes place by the expert touching with the mouse the nodes in the hierarchy. Note how the expert enters the information by typing English. This input is mapped into a frame structure and from there a program transform it into the OPERA data structures.

System Knowledge Organized into a Static Hierarchy

In order to further achieve our goal of helping the expert transfer his/her knowledge to the expert system, we have built a hierarchy which contains structural knowledge about the systems and devices. We have called this hierarchy the *static hierarchy*, because the knowledge in it can not be changed by the expert. The domain expert does not know that this hierarchy even exists. The interface uses this hierarchy to help the expert enter information. For instance, suppose that a domain expert is ready to enter information about the diagnostic advisory of a specific error, then, automatically, the interface displays in one of the windows all the diagnostic categories. The expert can then click with the mouse in one of the categories, and the interface will display all subcategories within that category. Now, the expert may decide to choose one of those categories by clicking with the mouse. If he does so, that category will be automatically inserted in the text he/she is typing.

We can establish an analogy between these two types of hierarchies, which we have called *experiential* and *static*, and the hierarchies of a diagnostician and a physiologist, respectively. Diagnosticians are trouble shooters. They organize their knowledge on the basis of experiences and cases. Physiologists have a deep knowledge about the organs and functions of the human body from a structural and causal point of view. Of course, there are situations in which a

diagnostician would like to know more about the function of organs, and they need to consult books during the diagnosis of a case. This is precisely the aspect which we are capturing in our interface. The knowledge in the static hierarchy is always ready to help the domain expert by bringing up relevant structural information about devices and programs.

The implementation of these two hierarchies has been done by constructing a unified data structure for all information. This structure facilitates ease of retrieval and updating of data, avoids redundant data and supports multiple access paths to information.

Current Effort

Our work will concentrate on the following areas: the testing of the present interface with domain experts and the analysis of the language used by domain experts. As result of this testing, the interface will require some changes. We do not anticipate major changes, but only small additions and refinements to the software components which are already implemented. In parallel with the testing process, we will be working on analyzing the English language entered by domain experts. This analysis will focus on two aspects: (a) uniformity of the language and (b) ambiguity. Presently, diverse experts use very different expressions to describe identical advisories and causes. In some cases, this does not pose any problem, but there are cases in which it is not clear that they are referring to the same situation. Consequently, we are going to design algorithms to uniformize as much as possible the language used by domain experts in describing the errors. Another aspect of our investigation will be to analyze the experts' language looking for ambiguity in the sentences. Our goal will be to investigate *lexical ambiguity* and the ambiguity of complex nouns. Syntactical and semantic ambiguity will not be part of our study during the first the quarter, although we may look into it during the second quarter, depending on our progress in analyzing lexical ambiguity. After this analysis is completed, our goal will be to help the expert define the terms as precisely as possible by pointing out ambiguous expressions and terms. The progress in this area, of course, will depend on the testing of the interface. If the current software does not require major modifications we plan to start

implementing some of the ambiguity algorithms during the first quarter of the second year.

Appendix A

Natural Language Knowledge Acquisition Systems

**UCF Dr. Fernando Gomez
Richard D. Hull
Clark R. Karr**

**Grumman R. Bruce Hosken
William Verhagen**

UCF Dept. of Computer Science NASA Grant 11-28-204

Presentation Agenda

- Introduction
- Plan and Schedule
- Summary of Work through last Review
- Current Effort
- Remaining Tasks

ORIGINAL PAGE IS
OF POOR QUALITY

UCF Dept. of Computer Science NASA Grant 11-28-204

Introduction

Purpose and Task

- **Purpose - To capture human expertise.**
- **Task - To create a natural language Interface whereby domain experts can transfer their knowledge to first generation expert systems.**

ORIGINAL PAGE IS
OF POOR QUALITY

UCF Dept. of Computer Science NASA Grant 11-28-204

Introduction

Typical Error Seen By Domain Expert

FEP 141 (\$\$\$\$) MICROCODE DID NOT RECEIVE AN ACKNOWLEDGE SIGNAL FROM THE I/O ADAPTER,
DATA ACQUISITION HAS BEEN INHIBITED MICAS=\$\$\$\$, NSB=\$\$\$\$

* TERMINAL ERROR FOR THE GSE FEP. THE I/O ADAPTER DID NOT SEND AN ACKNOWLEDGE
SIGNAL

TO THE MICROCODE DURING THE OPERATION INDICATED BY MICAS.

Probable Cause(s):

1. I/O Adapter failed.
2. GSE Option Plane failed.
3. I/O Adapter port on the 4-port controller failed.

Operations Advisory:

1. Halt the CPU, and record CPU registers. Push CPU through recovery.
2. If redundant FEP hasn't taken over, configure another FEP, or \$CLAI existing FEP again.
3. \$SPRCVE.
4. If redundancy isn't available, and original FEP fails to \$CLAI, then troubleshoot per following diagnostic advisory.
5. Lookup the MICAS in the microcode listings, and verify the operation being executed at the time of the anomaly.

Diagnostic Advisory:

1. \$DPLORT LI 5.
2. SEQ FEPID1, if errors occur I/O Adapter thumbin may assist troubleshooting.
3. GSE M02.
4. SEQ FEVTR1 (loop T/R via RCVS).

Introduction

OPERA Representation of Message Data

**** MSG-CAUSES: ****

- ((FILTERS) 1. FEP I/O Adapter failed.)
- ((FILTERS) 2. GSE Option Plane failed.)
- ((FILTERS) 3. I/O Adapter on 4-port controller failed.)

⋮

**** OPS-ADVISORY: ****

- ((FILTERS) 1. Halt the CPU, and record CPU registers. Push CPU through recovery.)
- ((FILTERS) 2. If redundant FEP hasn't taken over, configure another FEP...

⋮

**** DIAGNOSTIC-ADVISORY: ****

- ((FILTERS) 1. \$DPLORT LI 5.)
- ((FILTERS) 2. SEQ FEPID1, if errors occur I/O Adapter thumbin may assist...)

⋮

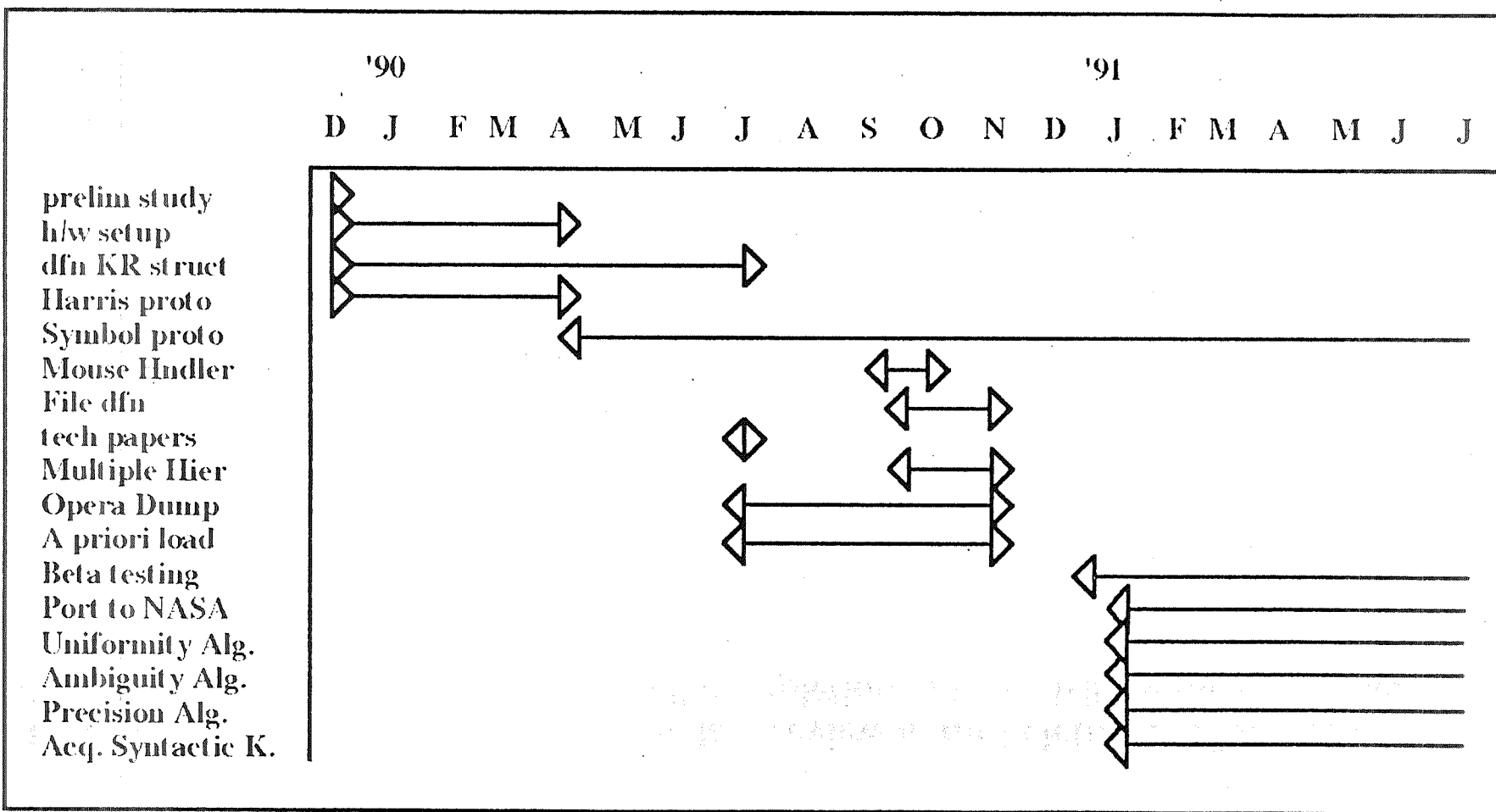
(Message Text and Other Information used by OPERA in maintaining the Error Message Knowledge Base.)

2.3

ORIGINAL PAGE IS
OF POOR QUALITY

Plan and Schedule

Schedule

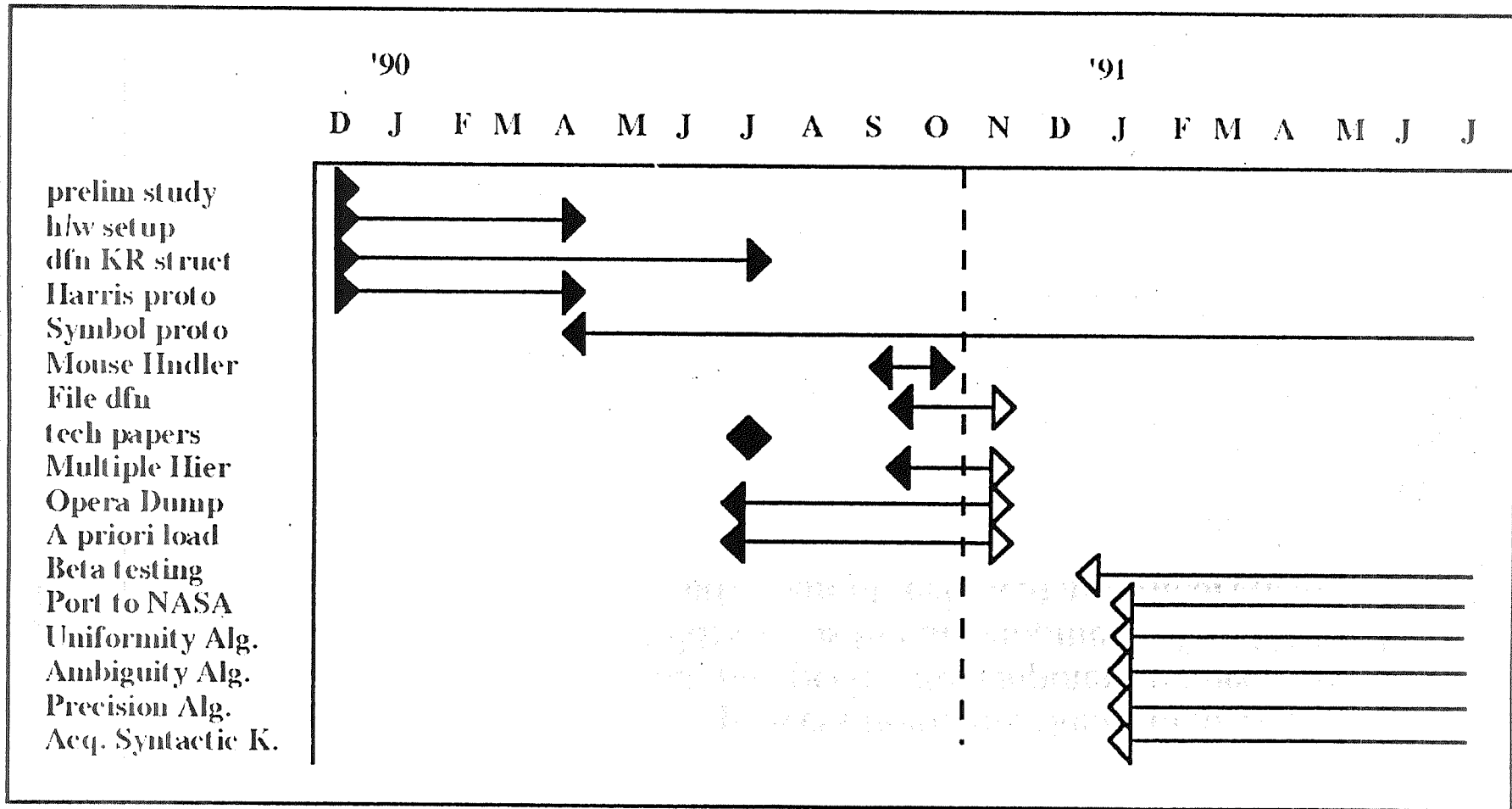


ORIGINAL PAGE IS
OF POOR QUALITY

UCF Dept. of Computer Science NASA Grant 11-28-204

Plan and Schedule

Schedule as of Last Review

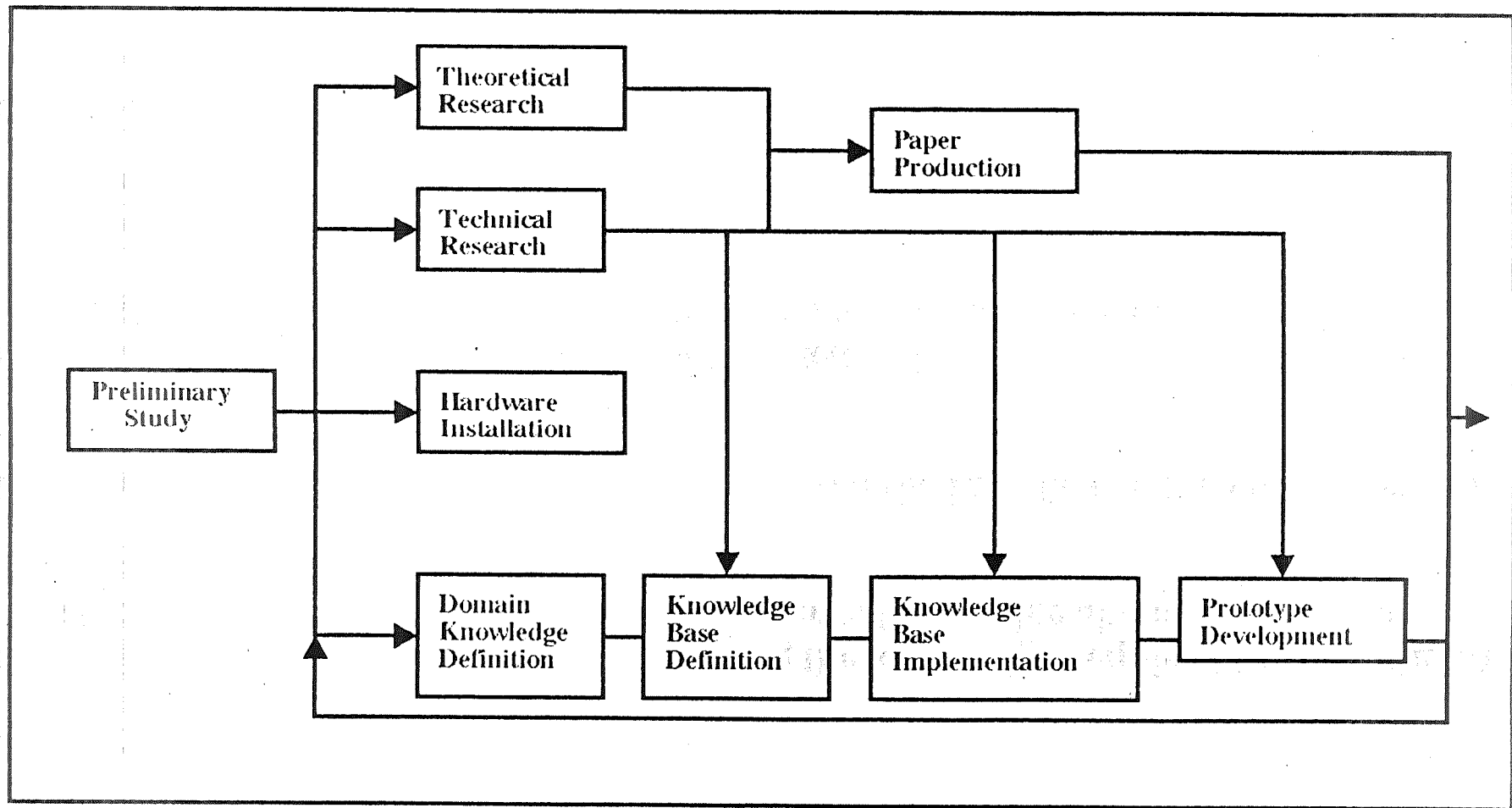


ORIGINAL PAGE IS
OF POOR QUALITY

UCF Dept. of Computer Science NASA Grant 11-28-204

Plan and Schedule

Plan



ORIGINAL PAGE IS
OF POOR QUALITY

UCF Dept. of Computer Science NASA Grant 11-28-204

Current Effort

Specific Tasks

- **File definitions for a priori knowledge.**
- **Routines needed to dump data acquired by the interface in OPERA format.**
- **Multiple hierarchies (static and expert) existing within a common framework.**
- **Data load program for the knowledge engineer.**
- **Alpha testing of interface system.**

ORIGINAL PAGE IS
OF POOR QUALITY

Current Effort

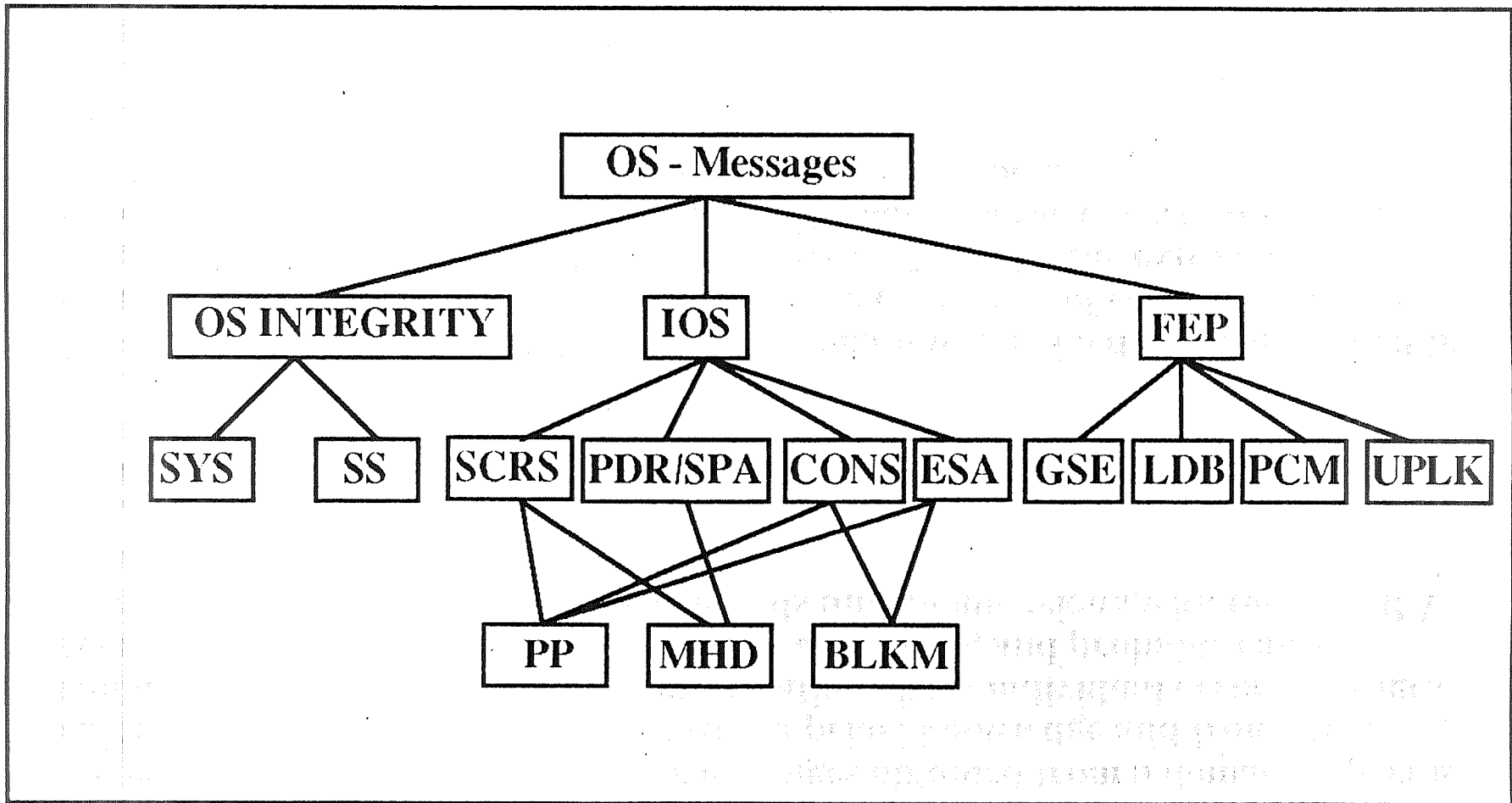
Static Hierarchy

- Contains a priori information
 - operational advisories
 - diagnostic advisories
 - probable causes
- Information visible to expert for selection during interview process.
- Not changable by domain expert
- Organization specified by Knowledge Engineer

ORIGINAL PAGE IS
OF POOR QUALITY

Current Effort

Static hierarchy



ORIGINAL PAGE IS
OF POOR QUALITY

UCF Dept. of Computer Science NASA Grant 11-28-204

Current Effort

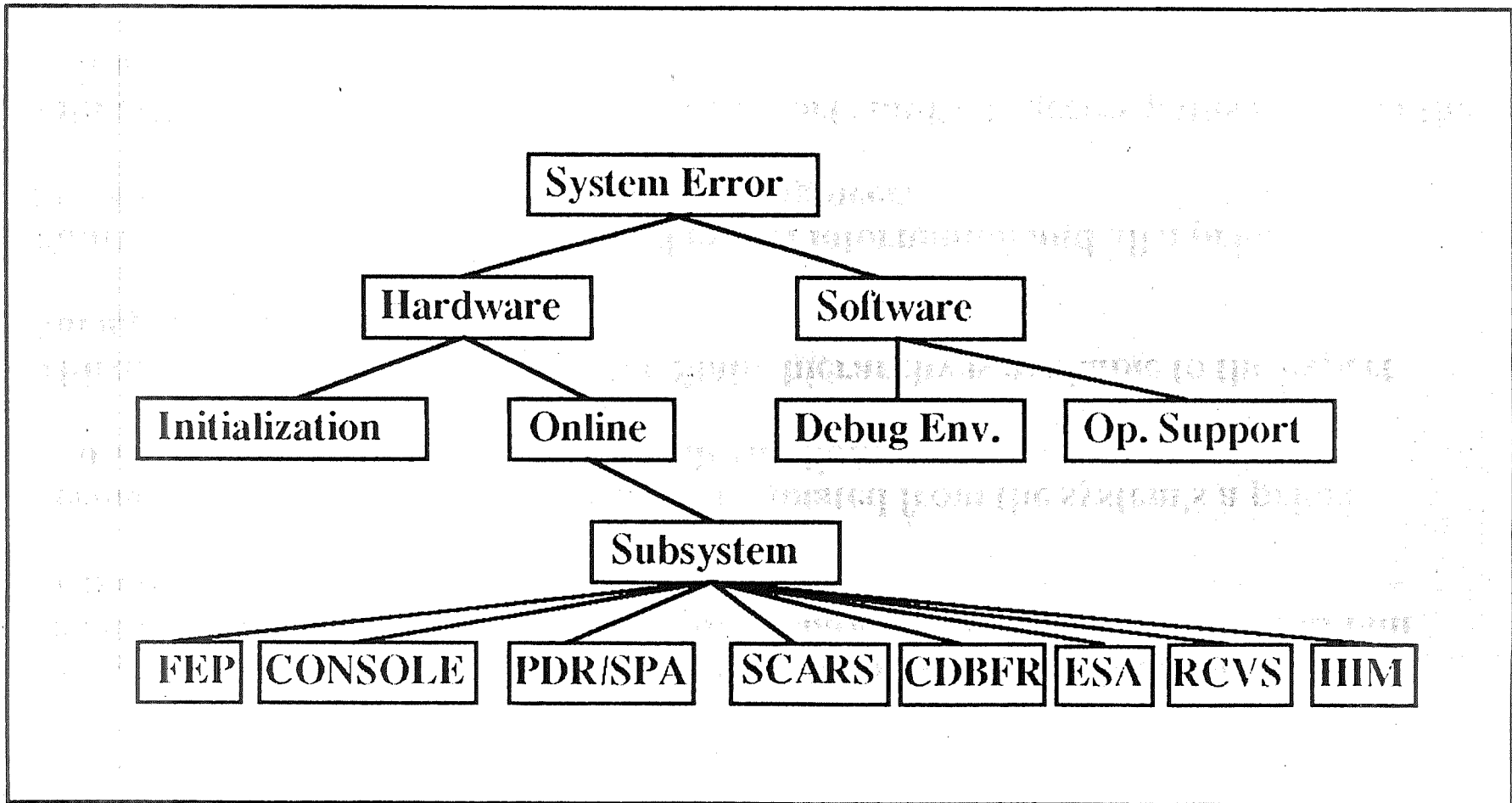
Expert Hierarchy

- Contains expert information
 - Knowledge specific to error message
 - operational advisories
 - diagnostic advisories
 - probable causes
- Selected information obtained from Static hierarchy.
- Changable by domain expert
 - Organization of hierarchy
 - Contents of nodes within hierarchy

ORIGINAL PAGE IS
OF POOR QUALITY

Current Effort

Expert Hierarchy



UCF Dept. of Computer Science NASA Grant 11-28-204

Current Effort

Underlying Knowledge Structure

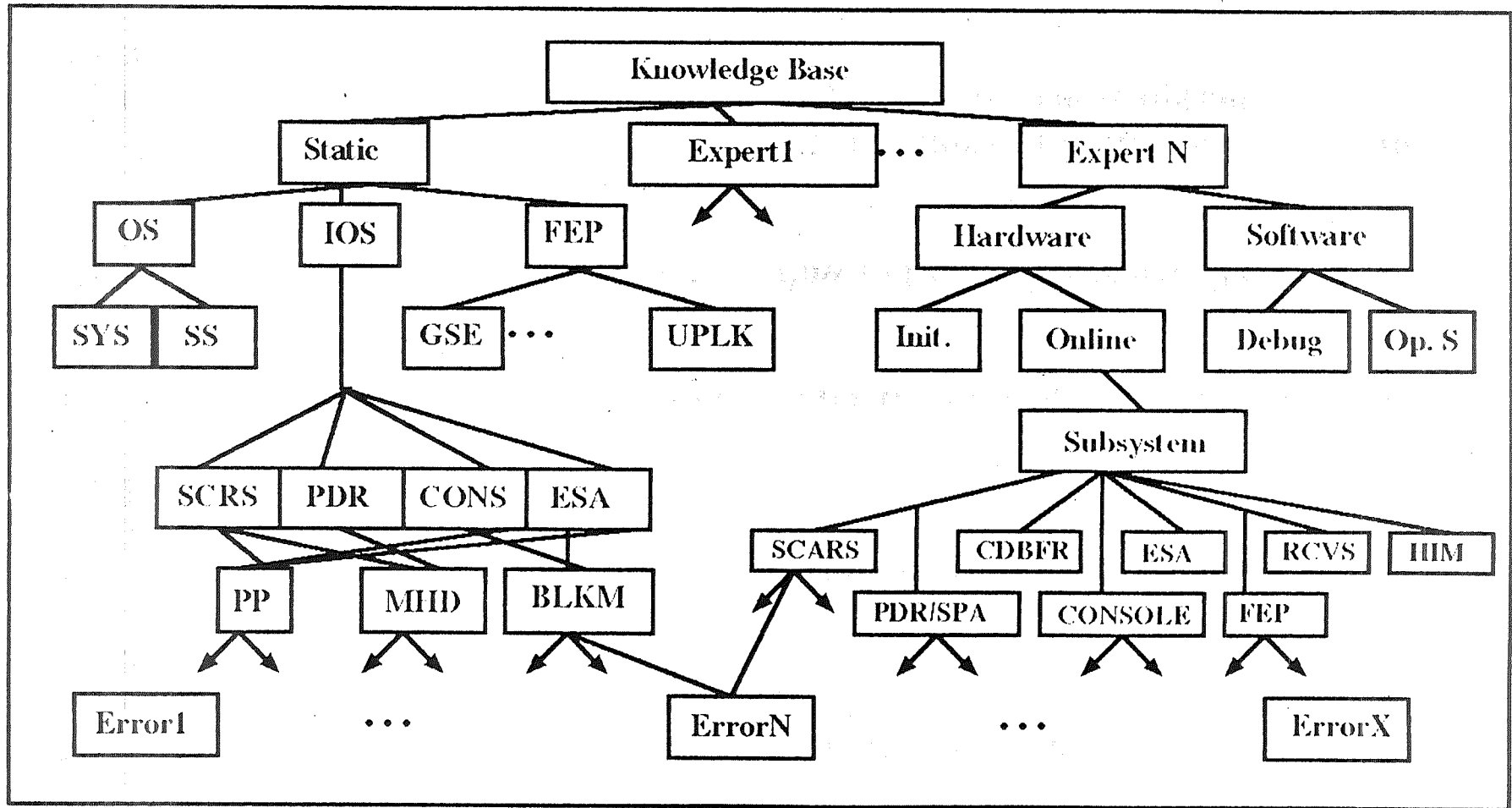
- **Single, unified data structure for all information.**
 - **Ease of retrieval and update of data.**
 - **Non-redundant data.**
- **Isolation of individual expert's knowledge.**
- **Static knowledge available to expert.**
- **All information available to Knowledge Engineer.**
- **Support multiple access paths to information.**

ORIGINAL PAGE IS
OF POOR QUALITY

UCF Dept. of Computer Science NASA Grant 11-28-204

Current Effort

Underlying Knowledge Structure



Opera Knowledge Acquisition Interface

Clear Screen
Help
Show Message

Decode Status Regs
Load Hierarchy
Static Hierarchy

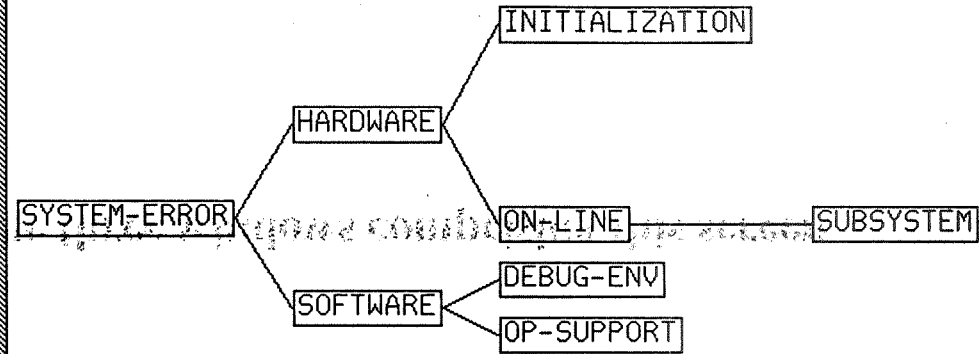
Delete Message
Preload Message
UnDelete Message

Describe Msg
Restructure Hierarchy

Edit Message
Save Hierarchy

Generate Dump
Show Hierarchy

Expert Last Name: hull
Expert First Name: r



ORIGINAL PAGE IS
OF POOR QUALITY

Opera Knowledge Acquisition Interface command: Load Hierarchy
Opera Knowledge Acquisition Interface command:

UCF Dept. of Computer Science NASA Grant 11-28-204

Message to Describe: 135-1

NIL

MESSAGE: 135-1

Is the category containing 135-1 in the hierarchy? (Y or N) Yes

What kind of message is 135-1? FEP-MSG

What type of message is 135-1

Enter Terminal or Non-Terminal: Terminal

Probable Causes:

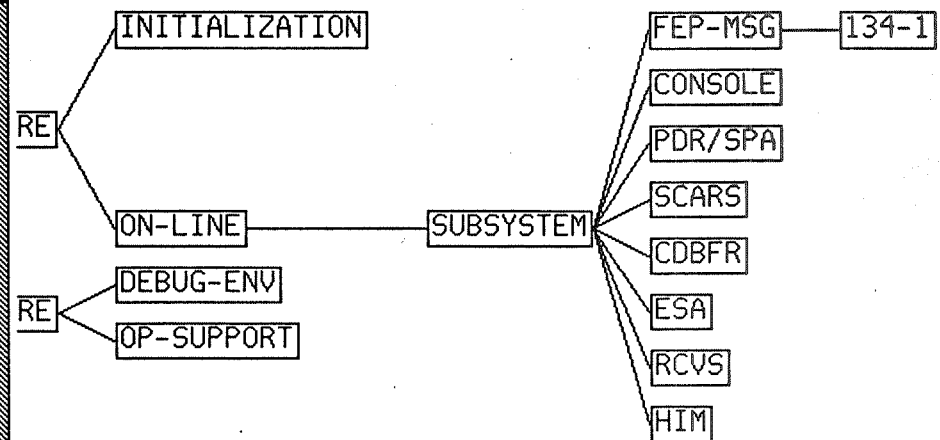
- 1: HIM Control Card failed.
- 2: HIM Master Control Card failed.
- 3: Transmitter/Receiver in the HIM failed.
- 4: GSE FEP T/R failed.
- 5: Ground Data Bus problem.
- 6:

Operational Advisory:

- 1: Gather *HARD COPY* output of \$HWMON and \$SYS pages for analysis.
- 2: Halt the CPU, record the CPU registers and perform a Recovery Dump.
- 3: Reload and reinitialize the failed FEP using \$CLAI (Conn on Load and Initialization) program.
- 4:

Diagnostic Advisory:

1:



Diagnostic Categories

SEQ FEPI, FEP I/O Adapter Diag
SEQ CDBK, Master Timing Unit Diagnostic
SEQ CP1, CPU DIAGNOSTIC PART 1
SEQ CP2, CPU DIAGNOSTIC PART 2
SEQ CP3, CPU DIAGNOSTIC PART 3
SEQ MEM, MEMORY DIAGNOSTIC
SEQ FPD, FLOATING POINT DIAGNOSTIC
SEQ SPI, SYSTEM PROTECT DIAGNOSTIC PART 1
SEQ SP2, SYSTEM PROTECT DIAGNOSTIC PART 2
SEQ OPD, OPTION PLANE DIAGNOSTIC
SEQ TIM, INTERVAL TIMER DIAGNOSTIC
SEQ MSM, 64K SEMI-CONDUCTOR MEMORY
SEQ CCPTST, CDBFR COMM. PROCESSOR TEST
SEQ CDBB, MICRO BUFFER DIAGNOSTIC TESTS
SEQ CDBM, CDBF DIAGNOSTIC, HEX

Message to Describe: 135-1

NIL

MESSAGE: 135-1

Is the category containing 135-1 in the hierarchy? (Y or N) Yes

What kind of message is 135-1? *FEP-MSG*

What type of message is 135-1?

Enter Terminal or Non-Terminal: Terminal

Probable Causes:

- 1: HIM Control Card failed.
- 2: HIM Master Control Card failed.
- 3: Transmitter/Receiver in the HIM failed.
- 4: GSE FEP T/R failed.
- 5: Ground Data Bus problem.
- 6:

Operational Advisory:

- 1: *Gather HARD COPY output of \$HWMON and \$SYS pages for analysis.*
- 2: *Halt the CPU, record the CPU registers and perform a Recovery Dump.*
- 3: *Reload and reinitialize the failed FEP using \$CLAI (Comm on Load and Initialization) program.*
- 4:

Diagnostic Advisory:

- 1:

Diagnostic Categories

SEQ FEPI, FEP I/O Adapter Diag
SEQ CDBK, Master Timing Unit Diagnostic
SEQ CP1, CPU DIAGNOSTIC PART 1
SEQ CP2, CPU DIAGNOSTIC PART 2
SEQ CP3, CPU DIAGNOSTIC PART 3
SEQ MEM, MEMORY DIAGNOSTIC
SEQ FPD, FLOATING POINT DIAGNOSTIC
SEQ SP1, SYSTEM PROTECT DIAGNOSTIC PART 1
SEQ SP2, SYSTEM PROTECT DIAGNOSTIC PART 2
SEQ OPD, OPTION PLANE DIAGNOSTIC
SEQ TIM, INTERVAL TIMER DIAGNOSTIC
SEQ MSM, 64K SEMI-CONDUCTOR MEMORY
SEQ CCPTST, CDBFR COMM. PROCESSOR TEST
SEQ CDBB, MICRO BUFFER DIAGNOSTIC TESTS
SEQ CDBM, CDBF DIAGNOSTIC, HEX

SEQ FEPI, FEP I/O Adapter Diag

TEST01 - Four-port Memory Checkout

TEST02 - Timer Checkout Using Self-test

TEST03 - Toggle I/O Register Checkout

TEST04 - Read/Write I/O Adapter Registers

Message to Describe: 135-1

NIL

MESSAGE: 135-1

Is the category containing 135-1 in the hierarchy? (Y or N) Yes

What kind of message is 135-1? *FEP-MSG*

What type of message is 135-1

Enter Terminal or Non-Terminal: Terminal

Probable Causes:

- 1: HIM Control Card failed.
- 2: HIM Master Control Card failed.
- 3: Transmitter/Receiver in the HIM failed.
- 4: GSE FEP T/R failed.
- 5: Ground Data Bus problem.
- 6:

Operational Advisory:

- 1: *Gather HARD COPY output of \$HWMON and \$SYS pages for analysis.*
- 2: *Halt the CPU, record the CPU registers and perform a Recovery Dump.*
- 3: *Reload and reinitialize the failed FEP using \$CLAI (Comm on Load and Initialization) program.*
- 4:

Diagnostic Advisory:

- 1: Use *SEQ FEPI, FEP I/O Adapter Diag* and sub-test *TEST01 - Four-port Memory Checkout.*
- 2: Also use *SEQ CDBK, Master Timing Unit Diagnostic.*
- 3:

Predecessor Messages:

- 1:

Remaining Tasks

- Continuing definition and implementation of knowledge base.
- Beta Testing.
- Port Interface to NASA.
- Natural Language
 - Ambiguity of Natural Language
 - Uniformity of Natural Language
 - Precision of Natural Language

Appendix B

Knowledge Acquisition From Natural Language For Expert Systems
Based On Classification Problem-Solving Methods

Fernando Gomez
Department of Computer Science
University of Central Florida
Orlando, FL 32816
gomez@ucf.csnet

Abstract

It is shown how certain kinds of domain independent expert systems based on classification problem-solving methods can be constructed directly from natural language descriptions by a human expert. The expert knowledge is not translated into production rules. Rather, it is mapped into conceptual structures which are integrated into long-term memory (LTM). The resulting system is one in which problem-solving, retrieval and memory organization are integrated processes. In other words, the same algorithm and knowledge representation structures are shared by these processes. As a result of this, the system can answer questions, solve problems or reorganize LTM.

1. Introduction

Can a simple expert system consisting of a hundred rules be designed from a natural language description by a human expert? Or can an expert system be updated from a natural language description of some of its components? In this paper, we present an integrated and domain independent system which builds classification-hierarchies from texts, retrieves information from them and solves problems.

We have been investigating the relation between comprehension and problem-solving. Our investigation has been guided by the idea that problem-solving and comprehension are not two separate processes each one with its own knowledge representation structures, but that they spring from a common source and share the same knowledge representation structures. It is beyond the scope of this paper to discuss these issues in detail, rather we center our discussion in providing evidence on how certain kinds of expert systems can be built from a natural language description by a human expert. We will show that the problem solving method used by the expert system to solve problems is the same as that used by the comprehender to keep memory organized and answer questions requiring chains of inferences.

We can roughly characterize the domain of the expert systems to which we have applied our method as consultation tasks. For instance, the tasks of building an expert system to select a programming language, or select a physician, etc. Some of these tasks are described in (Boose and Gaines, 1987). We can further characterize the scope of our system by the problem-solving method used to find solutions to a problem. The problem-solving method is that of classification problem-solving (Gomez and Chandrasekaran, 1984; Clancey, 1986). Classification-problem solving consists of conceptualizing the domain of the problem-solver into a hierarchy of concepts and designing an algorithm which solves a problem by placing it in the hierarchy of concepts. In (Gomez and Chandrasekaran, 1984) it is stated:

Given these principles of organization of medical knowledge, the solution of a medical

case becomes a problem of taxonomic classification. It is similar to the problem of placing, say, a specimen of maple in a hierarchy of botanical concepts. It consists of identifying its superordinate and subordinate concepts.

The method to recognize problem solutions by placing them in a hierarchy of concepts was that of using clusters of production rules organized under the concepts in the hierarchy, called *specialists* (Gomez and Chandrasekaran, 1984). In this paper, (a) we will show how to automatically build the classification hierarchies from natural language, and (b) we will describe an algorithm which does not use production rules to place the problem solutions in the hierarchy.

Among the knowledge acquisition systems which have been developed, ETS (Boose, 1984) and AQUINAS (Boose and Bradshaw, 1987) are the closer to our approach in the sense that these systems are domain independent and are based on building classification hierarchies for the solution of the problems. However, the differences are remarkable since we are not using elicitation techniques to build the classification hierarchies, but we are building them from natural language descriptions. Also, we are not translating the expert's knowledge into production rules, but we are using an general algorithm to produce solutions from the constructed hierarchies. Nano-KLAUS (Haas and Hendrix, 1980) was an earlier natural language acquisition system. However, their system is not targeted to the construction of expert systems, and it does not deal with the issues which are essential to our approach. The differences from other knowledge acquisition systems, MORE (Kahn, Nowlan and McDermott, 1985), MOLE (Eshelman, Ehret, McDermott and Tan, 1987) and SALT (Marcus, 1987), are more striking since these systems do not use classification problem-solving and presuppose domain knowledge.

This paper is organized as follows. In part 1 consisting of sections 2 and 3, we briefly explain some of the ideas on which our method is based, and in part 2 consisting of sections 4, 5, 6, 7 we describe how these ideas are applied to the construction of expert systems.

2. Background

For the last few years, we have developed a model of comprehension of elementary scientific texts or expository texts and a program which embodies the model, called SNOWY, (Gomez, 1985; Gomez and Segami, 1989a, 1989b). The comprehender of our model is a reader who is unfamiliar with most of the concepts which he/she is reading about. When SNOWY starts reading a scientific paragraph its domain knowledge consists only of a bare hierarchy of concepts. As SNOWY reads, new concepts are integrated in the right place in the hierarchy and new relations about already known concepts are learned. For instance, after reading the passage below with no knowledge about hormones and insulin, SNOWY acquires the concepts *hormones*, *insulin*, and *level of sugar in the blood*.

Hormones are chemicals produced by animals. Hormones control the activities of cells.
Insulin is a hormone produced by the pancreas. Insulin controls the level of sugar in the blood by making cells take in sugar.

SNOWY does not only acquire those concepts, but it also builds a classification hierarchy. For instance, the concept *insulin* is classified as a subconcept of *hormone produced by the pancreas*; this, as a subconcept of *hormones*; this, in turn, as a subconcept of *chemical*

produced by animals and, finally, this last one as a subconcept of *chemicals*. In contrast to knowledge-intensive models of comprehension which understand by framing the input in rich pre-built structures, SNOWY understands by using *formation* rules which build conceptual structures from the logical form of sentences. These conceptual structures are then passed to a *recognizer* algorithm which checks if those concepts exist in long-term memory (LTM). Finally, those concepts which the recognizer algorithm fails to recognize are passed to an integration algorithm which decides where to integrate those concepts in LTM. We have called these three phases concept formation, concept recognition and concept integration.

Although SNOWY is a knowledge-lenient model of comprehension, this does not mean that SNOWY does not possess any knowledge when it starts reading a text. Besides the linguistic knowledge which allows SNOWY to parse and disambiguate sentences (Gomez, 1988), SNOWY possesses a set of categories organized in a hierarchy, a set of primitive relations for action and descriptive verbs, which are used to build the logical form of the sentence, and a set of formation rules which build the LTM conceptual structures from the logical form of the sentences. Figure 1 contains a sample of the *a priori* hierarchy. The hierarchy is divided into physical things and ideas, as shown in Figure 1. The portion of the hierarchy which is shown in Figure 1 is strictly hierarchical. However, when the system begins to acquire new concepts, it builds a "tangled" hierarchy as described in the sections below.

As new sentences are typed, the system builds the representation of the new concepts and relations and inserts them into the hierarchy at the appropriate places. For example, for the sentence "germs are animates," the system adds the following to LTM:

(germ (is-a (animate)))

(animate (classes-of (germ)))

Before describing how to build these hierarchies, we need to explain briefly the knowledge representation structures used by our system. Three kinds of representation structures may be part of LTM: object-structures, action-structures, and event-structures. For purposes of this

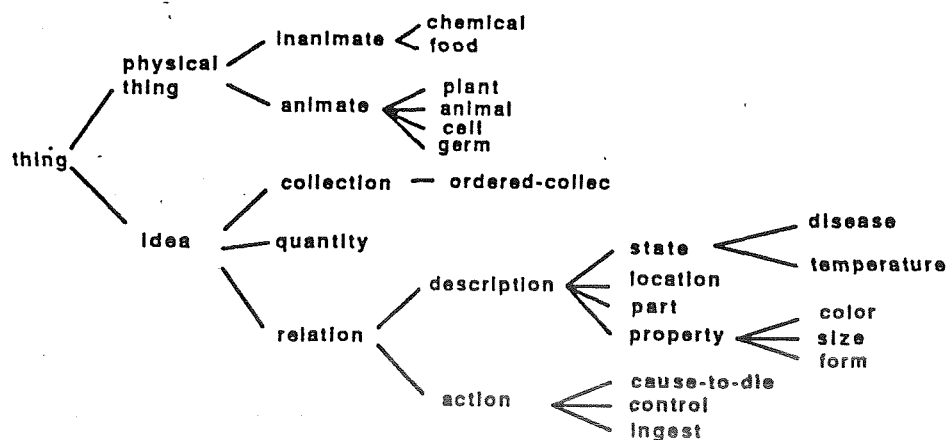


Figure 1: Partial content of the initial hierarchy of concepts in LTM

paper, we need only to describe object-structures and action-structures; event-structures are explained in (Gomez, 1986). All these structures are variants of the frame notation. An object-structure contains knowledge about physical or abstract objects and has slots called conceptual relations or predicates. Two kinds of conceptual relations can be distinguished: those which describe objects and those that attribute actions to physical things. This distinction corresponds to the distinction between descriptive and action verbs in natural language (Gomez, 1982). "Hormones are chemicals" and "all centipedes are slim and have many legs" are descriptive sentences, while "antibiotics kill germs" and "cells take in sugar" express actions performed by physical things. Descriptive attributes are also represented following the frame notation; for instance, the knowledge in the sentence about centipedes is represented as:

```
centipede
  (form (slim (q1 (all))))
  (has-body-part (leg (q1 (all)) (q2 (many))))
```

The slot *q1* contains the quantifier of the concept described by the object-structure, and the slot *q2* contains the quantifier of the second argument of the relation. Conceptual relations denoting actions are represented in the *object-structure* as:

```
relation (a1) (if the relation is monadic)
```

If the relation takes two arguments or more it is represented as:

```
(relation (concept1 a1))
```

where *concept1* is the second argument of the relation. If the same relation is true of two or more concepts, as is the case of the relation underlying the sentence "Penguins live in the sea and in the land," the relation is represented in the *object-structure* as:

```
(relation (concept1 a1) (concept2 a2) ... (concepti ai))
```

The terms *a1*, *a2*, ... *ai* stand for the names of the conceptual relations. The structure which represents the conceptual relation itself is called an *action-structure*. For example, the action-structure representing the conceptual relation in the sentence "antibiotics kill germs" looks like:

<i>a1</i>	(inst <i>a1</i> cause-to-die)
(args (antibiotics) (germs))	(actor <i>a1</i> antibiotics)
(pr (cause-to-die))	(object <i>a1</i> germs)
(q1 (?))	(quantifier-subject <i>a1</i> unknown)
(q2 (?))	(quantifier-object <i>a1</i> unknown)

Of these two representations, we will use the one on the left side. The one on the right side is a slot-assertion notation (Charniak and McDermott, 86). The slot *args* contains the arguments of the relation. If the relation is monadic, the slot *args* will contain one concept. If the relation is diadic (as in this case), the slot *args* contains two concepts, and so on. The first concept in the slot *args* is always the actor of the action verb and the second concept is the object. The slot *pr* contains the relation. The slot *q1* contains the quantifier of the first argument, and the slot *q2* has the quantifier of the second argument of the relation. If the relation is triadic, there will be a *q3* slot for the quantifier of the third argument. The representation of "antibiotics kill germs" will be done by using the following object-structures and action-structure:

antibiotics germs
 (cause-to-die (germs a1)) (cause-to-die:by (antibiotics a1))

a1
 (args (antibiotics)(germs))
 (pr (cause-to-die))
 (how (?))
 (q1 (?))
 (q2 (?))

Every concept which fills a case (actor, object, destination, source, etc.) in the sentence is indexed independently. In the above example, both the actor, *antibiotics*, and the object, *germs*, are indexed using an object-structure. Note how the concepts *germs* and *antibiotics* point to the same a1-structure, so that the question "what kills germs"? can be answered. The question mark in the slots *q1* and *q2* means that the sentence does not specify the quantifiers. If the sentence had said "all antibiotics kill germs," then the content of the slot *q1* would have been *all*. If the sentence had said "most antibiotics kill germs," the slot *q1* would contain *most*. Figure 2 contains the representation of "the human body kills germs by producing antibodies."

3. The Detection and Construction of Classification Hierarchies

The importance of explicit classification has been widely recognized and described by cognitive scientists working in reading. In explicit classification, the classes are clearly introduced and defined as in:

There are two kinds of mammals: terrestrial mammals and aquatic mammals. Aquatic mammals include the carnivorous and the omnivorous. The carnivorous includes the seals and walruses. The omnivorous includes the whales and etc.

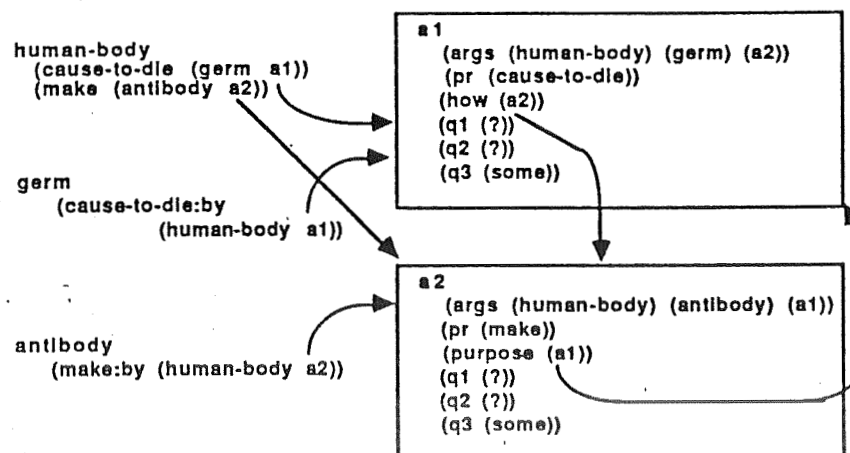


Figure 2: The representation of "the human body kills germs by producing antibodies"

In contrast to this type of classification, we will show that if a program is to acquire knowledge by reading texts it needs to classify in many instances in which classification is not apparent *prima facie*. Underlying classification is a more basic phenomenon than that which can be observed in other forms of classification, such as generalization-based classification or conceptual clustering. Furthermore, it is essential for constructing a problem solver from an English description of the task by a human expert, since the system will detect the classification hierarchies even when the expert does not see them. In the next sections, we will show that classification hierarchies underly two frequently used sentential structures: restrictive relative clauses and existentially quantified sentences.

3.1 Learning Concepts Denoted by Restrictive Relative Clauses

A first aspect of knowledge-acquisition is posed by the learning of concepts denoted by noun groups and restrictive relative clauses, e.g. "birds with long legs," "hormones produced by the pancreas," "programming languages which run in PCs," etc. The acquisition of these concepts involves their appropriate integration in the hierarchy, which in turn requires their classification by determining their parent and children nodes. For instance, consider the passage:

Birds with long legs live in swamps. They have a long bill to extract food from under water.

This text introduces the concept *bird with long legs* for which a name is not given. However, a unique name in memory needs to be created for that concept so that additional knowledge about this concept can be learned by storing this knowledge under the same node, and so this node can be linked to the other nodes in the hierarchy. The acquisition of the concept *bird with long legs* entails its classification so that it can be integrated appropriately in the hierarchy of concepts. If the system fails to classify the concept, it will not be able to answer such questions as "are there birds with long bills?" or "do birds with long legs have feathers?"

Restrictive relative clauses and complex noun groups are represented in LTM via object-structures with a dummy name (a gensym), and their representation is characterized by the presence of a *characteristic-features* slot (written *cf*), whose purpose is to identify the concept by describing the features that are characteristic of the concept. For example, when building the representation of the sentence "insulin is a hormone produced by the pancreas," the concept *hormones produced by the pancreas*, a subconcept of the concept *hormones*, is created with the following representation:

```
(x1 (cf (is-a (hormone)) (make:by (pancreas (q2 (?))))))
(hormones (classes-of (x1)))
```

Within the *characteristic-features* slot there is always at least an is-a slot, which indicates the parent node of the concept being learned. In this example, the concept *x1* has been created. *X1* stands for a subconcept of the concept *hormones*, as indicated by the is-a slot, and is further characterized by referring to those hormones made by the pancreas. The acquisition of the concept *x1* is completed by creating the slot *classes-of* in the concept *hormones*, and by filling it with *x1*. Had the slot *classes-of* existed in the concept *hormones*, then *x1* would be added to it. The newly-created concept is identified by the content of the *characteristic-features* slot and not by the dummy name *x1*. Thus, if the question "what do hormones

produced by the pancreas cause?" is asked, the system recognizes that the concept *hormones produced by the pancreas* was previously created and accesses the object-structure x1. (The slot q2 contains "?" because it is unknown if every pancreas makes hormones.)

The acquisition of concepts described by complex noun groups is done in a similar manner. The representation of *bird with long legs* proceeds by creating the concept *long leg* first, and then the concept *bird with long legs*. Both concepts are represented in Figure 3. This definition creates the concept x1, a subconcept of the concept *leg*. The concept *bird with long legs* is represented by x2. The *characteristic-features* slot in x2 says that x2 is a subconcept of *bird* characterized by having long legs. Because the noun group does not specify how many long legs x1 has, the quantifier slot q2, whose scope is x1, has a question mark in it. If the system has knowledge about birds, it fills this slot; otherwise, it will remain with a question mark until the system finds out more about birds.

3.2 Formation or Learning Rules

The mechanism we have described for acquiring concepts denoted by restrictive relative clauses and noun groups is based on *formation or, learning rules*. These are very general rules anchored in the verbal primitives, and in some instances in lexical items. In most cases these rules use knowledge in LTM and in the structure built by the parser.

The formation rules which form the concepts x1 and x2, which have been explained above, are part of the formation of noun groups. For instance, the antecedent of the rule which is fired in the example above is:

The head noun has the feature animate and is followed by the preposition "with," and the object of the preposition has the feature body-part.

Since the object of the preposition is also a noun group, the formation rules for the noun group are activated recursively to form the concept *long leg*. In this case, the antecedent of the rule activated is:

The head noun has the feature body-part and its modifier has the feature size.

The latter rule will form the concept x1 and will pass it to the former rule, which will form the concept x2.

This has been a very simple example to illustrate basic formation rules. In the next section, we explain how formation or learning rules are used to learn more complex classification hierarchies.

x1	x2
(cf(is-a (leg))(size (long)))	(cf(is-a(bird))(body-part(x1 (q2(?))))))
leg	bird
classes-of(x1)	classes-of(x2)

Figure 3: The Representation of *bird with long legs*

3.3 The Acquisition of Classification Hierarchies

As we have seen, a classification hierarchy may be indicated explicitly, as in the sentence "antibiotics are chemicals" or it may be stated in a more subtle way as in the phrase "hormones produced by the pancreas." There is yet a more complex way of expressing a complex hierarchy of concepts like the illustrated in the passage below:

(1) Not all animals which live in the sea are fish. Some are mammals.

These two sentences introduce three concepts: *animals which live in the sea*, *sea fish* and *sea mammal*. The last two concepts are subconcepts of the first one.

The learning of these concepts and the hierarchy which they form is done by means of a learning rule attached to *not all* and by one of the learning rules anchored in *are*. The first sentence is parsed into:

(subj (not all animals) rela (g00001) verb (are) pred (fish))

g00001

(subj (animals) verb (live) prep (in the sea))

The learning rule which builds the representation for restrictive relative clauses is activated. This rule activates the formation rule for the verb *live*. The structure built is:

x1	animal
(cf(is-a(animal)(habitat (sea))))	classes-of (x1)

The name of the concept which has been created replaces the concept modified by the relative clause in the main sentence, resulting in:

(subj (not all x1) verb (are) pred (fish))

The formation of the main sentence proceeds by activating the learning rules for *are*. The following rule fires:

if the subject is under the scope of an existential
quantifier or the negation of an universal quantifier then

- a. create a concept with a dummy name, say x2, and with
a characteristic-features slot containing the slots is-a (subj) and
is-a(pred)
- b. create another concept with a dummy name, say x3,
and with a characteristic-features slot containing the slots is-a (subj)
and is-a (?).

The terms *pred* and *subj* in the rules are variables containing the subject and the predicate of the sentence, respectively. This rule when applied to (1) gives:

x2	x3
(cf (is-a (x1)(fish)))	(cf (is-a (x1) (???)))

The application of the rule for restrictive relative clauses and the one for *are* has resulted in the creation of three concepts: the concept x1, which corresponds to *sea animal*, the concept x2, which corresponds to *sea fish*, and x3, which has been only partially specified.

Adverbs such as *almost*, *only*, and the the negation of universal quantifiers, e.g., *not all*, introduce links which connect the sentence where they appear to the next sentence. In order to solve these links, these lexical terms have attached to them formation rules. These rules are activated as part of the formation of the sentence which immediately follows the one which starts with *not all*, etc. These formation rules resolve the anaphoric reference of *some* in the second sentence and fill in the question mark in the is-a slot of the concept x3 above. The concepts and the hierarchy formed for the passage *Not all animals which live in the sea are fish. Some are mammals.* are found in Figure 4. The formation rules for *not all* do not only deal with this example, but also with examples such as:

Not all animals which live in the sea are fish. Whales are mammals.

In this case, the formation rule checks LTM to see if whales live in the sea. If so, it will replace the question mark in the concept x3 above with *mammal*, and create *whale* as a sub-concept of x3. However, even if there is no knowledge about *whales* in LTM, the system infers from the structure of the sentence that *whales* are sea animals.

3.4 The Acquisition of Concepts Denoted by Existentially Quantified Sentences

The acquisition of concepts introduced by sentences starting with an existential quantifier requires the construction of a classification hierarchy in the same manner as we have done for relative clauses. The sentence "some mammals live in the sea" introduces the concept *mammal which live in the sea*. However, the representation in Figure 5 is inadequate for the job. The problem with this representation becomes obvious if the next sentence is "these mammals are intelligent." Since the structure above has not created the subconcept *mammal which live the sea* there is no node in the hierarchy to integrate the knowledge that they are intelligent. The representation of the sentence "some mammals live in the sea" is given below:

x1 (cf (is-a(mammal)) (habitat (sea))) --- is-a ---> mammal

X1 represents the concept *animals which live in the sea*. The concept *mammal* will contain

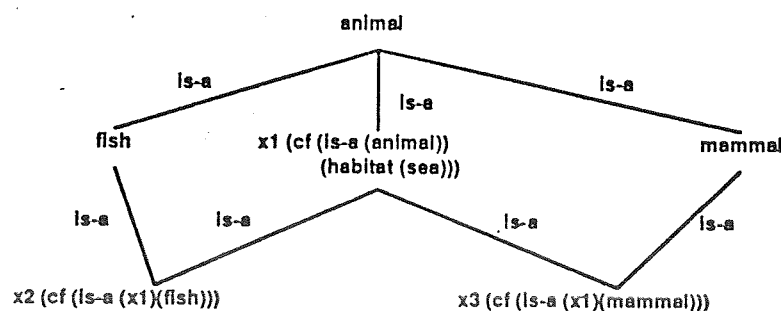


Figure 4: The hierarchy constructed for "Not all animals which live in the sea are fish. Some are mammals."

```

mammal      sea
habitat (sea a1)  habitat: by (mammal a1)

a1
args(mammal,sea)
relation (habitat)
q1 (some)
q2 (?)

```

Figure 5: Inadequate representation of "some mammals live in the sea."

the slot "classes-of (x1)." The knowledge expressed by the sentence "these mammals are intelligent" can now be integrated under the concept x1.

The role of classification is one of the most pervasive aspects in the structure of expository prose. As a final example consider the text below:

Some birds are unusual because they cannot fly. The emu, the ree, and the ostrich are such birds. The emu is an Australian bird. The ree lives in South America. The ostrich lives in Africa and runs very fast.

Some birds are unusual because they form a partnership with an animal. The oxpecker eats insects from the rhinoceros. The honey guide will lead a badger to a bees' hive.

```

-----
-----
-----
-----
-----
-----
-----
-----

```

The author found this text in a 6th grade comprehension exercise to teach children classification. The children were asked to write the names of birds on the dotted lines which follow the text. This text illustrates clearly that only through classification can the concepts and relations in the text be learned.

4. The Application of These Ideas to the Automatic Construction of Expert Systems

We will show in this section how these ideas can be applied to the construction of an expert system. The task which we will study is that of building a consultation system. We will use an example similar to that discussed in (Boose and Bradshaw, 1987), the selection of a programming language.

The expert is instructed to present his/her subject in a top down fashion describing first the most general concepts and then refining them until reaching the most concrete concepts which will form the tip nodes in the hierarchy. The paragraph below is just one way of how this

description may be accomplished. Many other are possible. After reading this paragraph, SNOWY builds the hierarchy of concepts shown in Figure 6.

There are several kinds of programming languages. Programming languages which manipulate symbols, called symbolic languages, are used for AI applications, while numeric languages are used for scientific applications. There are two main symbolic languages: Lisp and Prolog. Lisp is used for general AI applications, whereas Prolog is used when backtracking is needed. Common Lisp, Franz, Scheme and Golden Hill are Lisp languages. Franz runs in minicomputers and takes more than 3 megs of memory. Scheme is a good dialect of Lisp for PCs. It takes 0.8 meg of memory. Golden-Hill is also a version of Lisp for microcomputers, but it requires 2 megs of memory. There are several versions of Prolog, such as Quintus, Turbo and Mprolog. Of these, Turbo runs on PC's, and requires only 0.5 megs of memory. Numeric languages

The system proceeds by parsing the sentences. In the current implementation, the system can parse any sentence containing unknown nouns and adjectives, but verbs must be known to the parser for a successful parse to occur. If the parser fails then the system asks the user for a definition of those words which are unknown to the system. First of all, the system asks the user if the word in question is a verb (the user can check that by consulting a dictionary). If the word is a verb the system obtains the syntactical usage of the verb by activating an interface which allows users without a formal knowledge of English grammar to add new verbs to the parser. The system can make sense of many sentences containing words unknown to it. We postpone the discussion of how this is done until the next section. We briefly explain some of the formation rules used to construct the hierarchy in Figure 6. The first sentence does not create any structure. From the second sentence, the system creates the concepts *symbolic language* and *numeric language*, according to the techniques which we have explained for the formation of concepts denoted by restrictive clauses and noun groups. The *characteristic-features* slot of the concept *symbolic language* contains the conceptual relation "manipulate symbols." Note that from this moment on, the meaning of *symbolic language* is a

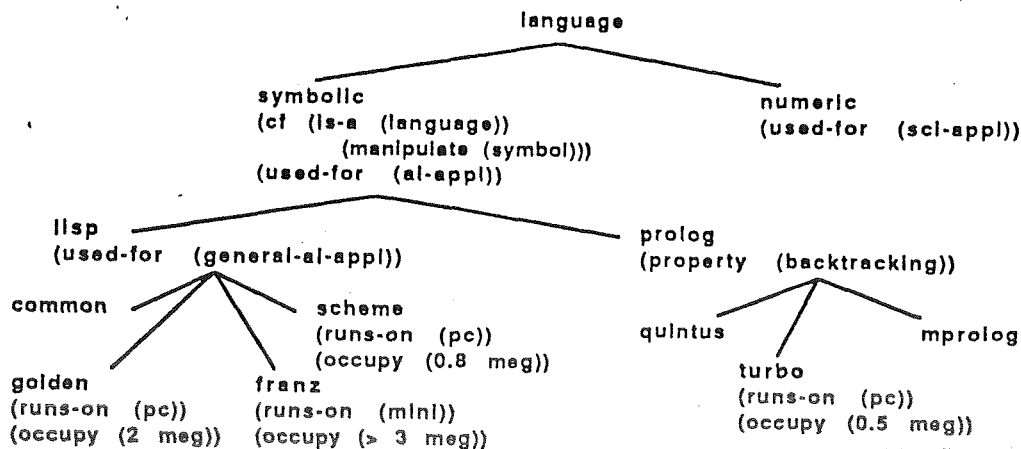


Figure 6: Hierarchy for the Text on Programming Languages

programming language which manipulates symbols. The formation of "symbolic languages are used for AI applications," is integrated by creating the descriptive relation "used-for (ai-application)" under the concept *symbolic language*. The next sentence creates two subconcepts of *symbolic language*: Lisp and Prolog. The next sentence differentiates the two languages. The formation rule which builds the LTM representation of "Prolog is used when backtracking is needed" constructs the following structure:

Prolog
(property(backtracking))

This is a general rule which forms all logical forms which fall under the schema:

<somebody> <use> <obj1> when/if/in case that <somebody> <need> <obj2>

into the structure:

obj1(property(obj2))

The rule is using the inference that if something, say *x*, is used when something else, say *y*, is needed then *y* is a property of *x*. This is necessary if requests such as "I need a symbolic language with backtracking" are to be answered by the consultation system.

5. Defining Unknown Words

One of the critical issues in the design of the system is how many concepts in the user's input need to be previously known by it in order for the system to understand the text and solve problems. If the knowledge acquisition system is going to be domain independent, then the system can not have *a priori* knowledge about concepts which are domain specific. For instance, the concept of "backtracking" in the sample text can not be part of the *a priori* concepts. However, if the knowledge acquisition system is going to deal with a concrete domain, then one can provide it with a rich set of domain dependent concepts. Since our goal is to build a domain independent knowledge acquisition system, the problem of unknown concepts becomes critical. The sample text can be processed by the present implementation without knowing anything about the following concepts: "programming languages," "symbols," "AI," "numeric languages," "backtracking," "Lisp languages," "meg" and "Lisp," Prolog," etc. If one asks the system "what is backtracking," this will reply by saying "I do not know what it is, but I know that it is a property of Prolog." The system's reply to a question is "I do not know" if the concept in the question is not linked to the concepts in the *a priori* hierarchy. For classification problem solving, the system does not need to have a deep understanding of what backtracking is. In any case, the decision of determining if a concept should be fully known to the system rests on the user. We will describe below some ways in which the user can achieve this.

Let us go back to the acquisition of the other concepts in the text. The mechanism for acquiring the concept "meg" differs from the acquisition of the other concepts in the paragraphs. The sentence in which the concept "meg" appears for the first time is "Franz ... takes more than 3 megs of memory." Let us assume that the system knows that memory is something where information is stored. The problem here is that when the system is going to form the final knowledge representation from the logical form of the sentence, it can not determine the

meaning of "takes" because it does not know the meaning of "meg." One of the meanings of "takes" is to occupy space or storage. A rule attached to "take" says: "if the object of "take" is unknown, but is numerically quantified then assume that the meaning of "take" is to occupy space." This rule will infer that meg is a unit of measurement and will ask the user to confirm its inference.

If a user deems that the system should have a concept classified in the hierarchy, or the system fails to build the representation of a sentence because some key concepts are missing, then the user can define these concepts by using the methods explained below.

Synonyms: This is the most linguistic and superficial of the methods. This method does not place concepts in the hierarchy. It is also the easiest to use, although, of course, it has limitations. For instance, the word "dialect of" in the middle of the paragraph may be unknown to the system. An easier way to define this word is to say that it is a synonym of "kind of," which the system clearly knows because it appears in the first sentence of the paragraph. Note that the words need not to be synonyms in all contexts, but just in the one the user is dealing with.

Classification: The best way to define a word is by providing the upper-concept of the concept denoted by the word to be defined. For instance, the word *AI* can be defined by saying "AI is a computer science." An inappropriate definition will be to say something like "AI tries to make computers intelligent," since this does not place *AI* in the hierarchy of concepts which forms the basis of SNOWY's understanding process. Likewise, the definition the system is expecting of *backtracking* is not a long and intricate one, which probably will be irrelevant to the expert system being built, but just something like "backtracking is a programming technique" which places *backtracking* in the hierarchy of concepts. We are extending the *a priori* hierarchy of categories to include a large set of concepts so that it will be easy to place new concepts in the hierarchy by classifying them.

Part-of Definitions: Some words may be defined by using a *part-of* relation. For instance, the word *cpu* or *computer memory* can be defined by saying that they are part of a computer.

Functional Definitions: We are studying short functional definitions of concepts from which the system can infer the *part-of* relation plus some other information. For instance, the concept *computer memory* above can be defined as: "memory is something to store information into a computer." From this, the system can infer that *computer memory* is a physical thing, part of a computer and is used to store information. The reason why this method of definition has not been incorporated in the system, yet, is because it opens the door for a user to give a definition too complex to be understood.

6. Problem-Solving

After SNOWY has read the paragraph of section 4, a user interested in a consultation about programming languages may, at this point, run a consultation test. For example, the user may say "I am interested in a language that runs on PC's and is good for ai applications." Given this inquiry, the task of the system is to find the languages that have the features "runs on PC's" and "used-for ai applications." The class of languages that have these features is represented as:

x1 (cf (is-a (language)) (runs-on (pc)) (used-for (ai-appl)))

If this concept were present in LTM, the question would be answered by simply retrieving its children. Since this is not the case, one way to find the answer to the question is to *classify* the concept *x1* ("languages that run on PC's and are good for ai") within the current hierarchy of concepts. This task is done by a *classifier algorithm*, which determines where in the current hierarchy a new concept, such as *x1*, would be located if it existed in LTM, that is, the *classifier* determines which of the concepts currently in LTM are the parents of *x1* and which ones are its children. We briefly illustrate the steps taken by the *classifier*, using the concept *x1* as an example. (See (Gomez and Segami, 1989a) for a detailed description of the algorithms in this section and a comparison to other classifier algorithms). Given the representation structure of *x1*, the concept within its *is-a* slot determines what part of LTM needs to be considered by the *classifier*. In this case, only the concept *language* or its children can be the parents or the children of *x1*. The *classifier* performs a traversal of selected nodes under the concept *language*. The concept in each visited node is compared with the concept *x1*. Depending on whether one is a subclass of the other, or there is no hierarchical relationship between them, the *classifier* decides which node to visit next. The actual comparison of concepts is performed by the *compare-classes* algorithm, which is based on the following idea. A concept *x2* is a subclass of a concept *x1*, if each characteristic-feature of *x1* is true of each entity in the class *x2*. In our example, concept *x1* is first compared with the concept *symbolic languages*. No hierarchical relationship is found between these two concepts. However, one of the *characteristic-features* of *x1*, "used-for ai applications," is found to be true of all the descendants of the concept *symbolic languages*, so that, when the nodes under *symbolic languages* are examined, only one more feature needs to be verified: "runs on PC's." After doing this, it is found that *scheme*, *golden*, and *Turbo* run on PC's. Then, the *classifier* returns the list "((language) (scheme golden turbo))" to indicate that the concept "languages that run on PC's and are good for ai" is a child of the concept *language* and that it has the children *scheme*, *golden*, and *Turbo*. The system, then, would reply to the user as follows: "the following languages run on PC's and are good for ai: scheme, golden, and turbo." If the user decides to narrow down his/her choices, he/she may either specify additional features, or formulate questions. For example, the following interaction may now take place:

user:

the language must run in a computer with less than 1 meg of memory

system:

scheme and turbo require less than 1 meg of memory

user:

what are the differences between scheme and turbo?

system:

scheme is a lisp dialect used for general ai applications, while turbo is a prolog dialect used for backtracking

The first sentence typed by the user simply adds one more requirement to the desired programming language. Since we have an initial set of solutions, the new requirement needs to be verified for the members of this set only. This reduces the possible solutions to two. The second sentence typed by the user is a question, which requires that a comparison be made

between the features of *Scheme* and *Turbo*. This is accomplished by traversing the hierarchy upwards, from the nodes corresponding to *Scheme* and *Turbo*. The traversal stops when the two paths cross each other.

The *classifier* is the component that allows SNOWY to maintain a correct organization of its memory. Each new statement entered by the expert may cause the insertion of a new concept in the hierarchy, or it may trigger a reorganization of the concepts in LTM. Let us suppose, for example, that the expert continues entering information to the system, by typing the following:

Some languages are procedural. Procedural symbolic languages are widely available.

The sentence "some languages are procedural" is integrated as explained in section 3.4. SNOWY's long-term memory configuration for the two sentences is depicted in Figure 7.

If the expert now types the statement: "Lisp is a procedural language," the concepts in LTM will be reorganized as shown in Figure 8.

Thus, in an inquiry of the form "a language that runs on PC's, is good for ai applications, and is widely available," only the Lisp dialects will be considered to be possible solutions.

As one can see the problem-solving method which we have illustrated uses only categorical knowledge (Szolovits and Pauker, 1978) and not probabilistic knowledge. It is our belief that probabilistic reasoning plays a minor role in most problem-solving situations if memory organization and indexing form part of the problem-solving method. In any case, certain aspects of probabilistic reasoning need to be included in our system. Experts use sentences

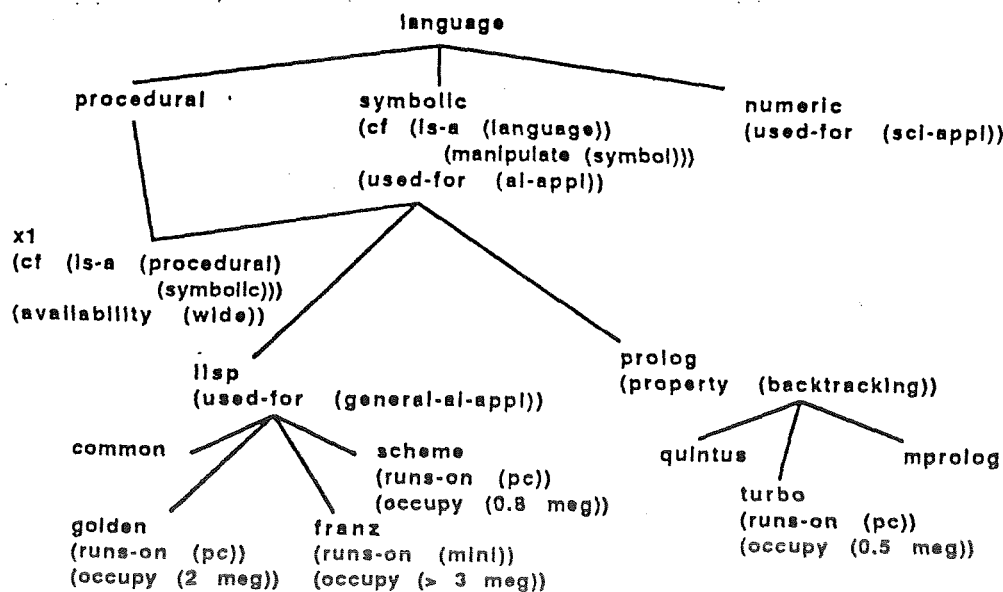


Figure 7: Organization of LTM after processing the passage "Some languages are procedural. Procedural symbolic languages are widely available."

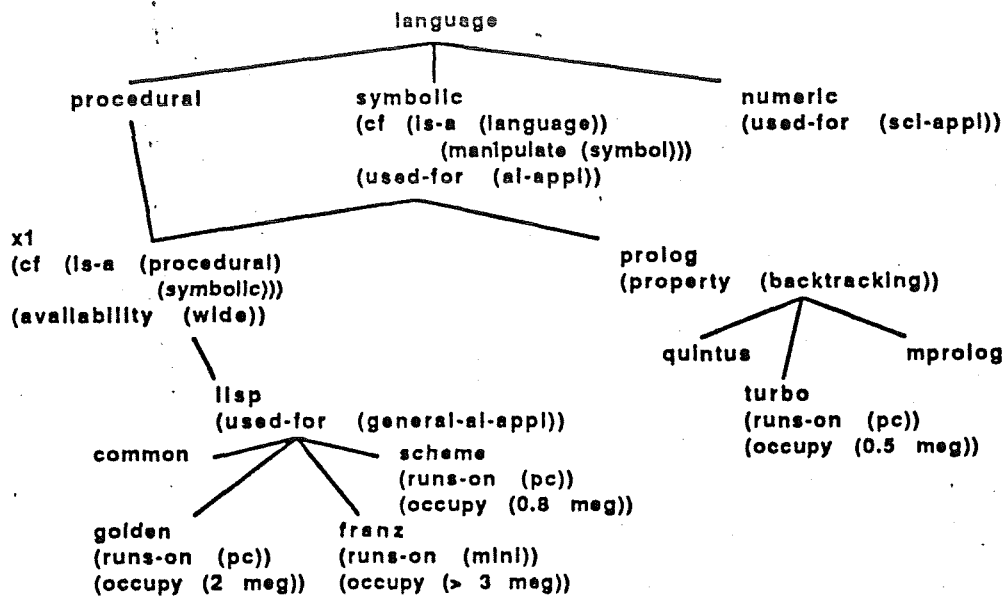


Figure 8: Reorganization of the concepts in LTM after processing the sentence "Lisp is a procedural language."

like this "Lisp is easier to learn than Pascal which in turn is easier to learn than Cobol." In cases like this, we need to ask the expert to rate Lisp, Pascal and Cobol with respect to the predicate *learn*. There are many ways in which this rating can take place and AQUINAS offers several ways which we plan to adopt in our system. Let us consider for the sake of the discussion that the expert produces the following rating: (Lisp (learn .9)) (Pascal (learn (.7))) (Cobol(learn (.4))). Then, the phrase "a symbolic language that is easy to learn" in the user's request "I am interested in a symbolic language which runs in a PC and is easy to learn," will be translated into a numeric value which may be greater than 5 according to the rating provided by the user for that predicate.

7. Discussion

In this paper, we have shown how a consultation system can be built from a natural language description by a human expert. It is our belief that the ideas and techniques described here can be used for the construction of expert systems for tasks which can be modeled using classification problem-solving. These tasks do not have to deal with consultation systems. They may be about diagnosis, design, etc. The relevance of the ideas presented in this paper go beyond the knowledge-acquisition area, since we have introduced a problem-solver which has striking differences from existing rule-based systems. In our system, organization and reorganization of memory, comprehension and problem solving are integrated processes. As a result, two important aspects of expert system technology become almost trivial: the updating of the knowledge base and interacting with the expert system. However, our approach is not without limitations. Let us first discuss the theoretical limitations. Although we believe that a

large class of problems and domains can be handled using our system, there are many domains which are beyond the reach of our method. The reason for that is that there are many aspects of the real world which can not be represented using the classification hierarchies we have presented in this paper. For instance, the functioning of complex systems such as the circulatory system, or the respiratory systems can not be adequately grasped using our present representation techniques. We are working in knowledge representation structures which will represent these systems from a point of view which will be adequate for comprehension and problem-solving (Gomez and Segami, 1989c).

There are many limitations to our present implementation. First of all, no expert has used yet our program to build a consultation system. All the consultation examples have been entered by graduate students or by the implementers of the system. Our system is a prototype which we can "demo" at any time, but that has not been used by any real user. This is clearly in contrast with AQUINAS which, according to the authors, has been used to build a variety of consultation systems. We are going to start to experiment with some real users to assess the real world capabilities of our system. Since SNOWY is able to acquire new concepts and new words, it can augment its lexicon by interacting with a user. The parser (Gomez, 1988) being used by SNOWY is a modular algorithm based on the notion of *syntactical usage of a word*, which allows a naive user without knowledge of English grammar to increase its syntactical coverage. We are writing an interface to our parser which will allow a user to add new verbs not presently in the parser. Yet, we are aware that there is a qualitative jump from testing a prototype by people who have certain familiarity with it to a real world system used by naive users.

We have presented a system and distinguished it from existing knowledge acquisition systems. Yet, our approach can be integrated with present knowledge acquisition tools and vice-versa, techniques in other systems can be incorporated in our system. We think that, in future developments of our system for real users, the elicitation techniques developed for ETS and AQUINAS can be very useful. In particular, we are thinking that in those cases in which the English of certain users becomes too hard to handle, SNOWY may fall back on elicitation techniques. Likewise, domain dependent knowledge acquisition tools can benefit from a system which is able to build classification hierarchies from natural language text.

References

- Boose, J. (1984). Personal construct theory and the transfer of human expertise. *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas.
- Boose, J. and Bradshaw, J. (1987). Expertise transfer and complex problems: using AQUINAS as a knowledge acquisition workbench for expert systems. *International Journal of Man-Machine Studies*, 26(1), 3-28.
- Charniak, E. and McDermott, D. (1984). *Introduction to Artificial Intelligence*. Addison-WesleyR, Reading, MA.

- Clancey, W. J. (1985). Heuristic Classification, *Artificial Intelligence*, 27, 289-350.
- Eshelman, L., Ehret, D., McDermott, J. and Tan, M. (1987). MOLE: a tenacious knowledge acquisition tool. *International Journal of Man-Machine Studies*, 26(1), 41-54.
- Gomez, F. (1982). Towards a Theory of Comprehension of Declarative Texts, in *Proceedings of the 20th Meeting of the Association of Computational Linguistics*, Toronto, Canada, 36-43.
- Gomez, F. (1985). A Model of Comprehension of Elementary Scientific Texts, in *Proceedings of the Workshop on Theoretical Approaches to Natural Language Understanding*, Halifax, Nova Scotia, 70-81.
- Gomez, F. (1986). Knowledge Representation for Understanding Expository Texts, Technical Report CS-TR-86, Department of Computer Science, University of Central Florida, Orlando, Florida.
- Gomez, F. (1988). WUP: A Parser Based on Word Usage, in *Proceedings of the 1988 IEEE Conference on Computers and Communication*, Scottsdale, Arizona, 445-449.
- Gomez, F. and Chandrasekaran, B. (1984). Knowledge Organization and Distribution for Medical Diagnosis, in W. Clancey and E. Shortliffe (Eds.), *Readings in Medical Artificial Intelligence*. Addison-Wesley, Reading, MA.
- Gomez, F. and Segami, C. (1989a). The Recognition and Classification of Concepts in Understanding Scientific Texts. *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 1, 51-77.
- Gomez, F. and Segami, C. (1989b). Classification-Based Inferences in Retrieving Information from a Database of Scientific Facts. In *Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications*, Miami, Florida.
- Gomez, F. and Segami, C. (1989c). Representing Systems as Event Hierarchies. Technical Report CS-TR-89-02, Department of Computer Science, University of Central Florida, Orlando, Florida.
- Haas, N. and Hendrix, G. (1980). An Approach to Acquiring and Applying Knowledge. *Proceedings of the National Conference on Artificial Intelligence*, Stanford, CA, 235-239.
- Kahn, G., Nowlan, S. and McDermott, J. (1985). MORE: an intelligent knowledge acquisition tool. *Proceedings of Ninth International Conference on Artificial Intelligence*, Los Angeles, California.
- Marcus, M. (1987). Taking backtracking with a grain of SALT. *International Journal of Man-Machine Studies*, 26(4), 383-398.
- Szolovits, P. and Pauker, S. (1978) Categorical and Probabilistic Reasoning in Medical Diagnosis. *Artificial Intelligence*, Vol. 11.

Appendix C

Finding and Learning Explanatory Connections from Scientific Texts

Fernando Gomez and Carlos Segami
Department of Computer Science
University of Central Florida
Orlando, Florida 32816

Abstract

A theory for detecting and learning the explanatory connections between sentences in scientific texts is presented. A program embodying the theory is also described. The knowledge in the program is organized around the notions of analytic and empirical knowledge. Analytic knowledge encompasses very general rules which are valid across any domain, while empirical knowledge includes rules whose validity is domain dependent. Examples of these rules and their representation are given.

1. Introduction

In the last few years, we have proposed a model of comprehension of scientific or expository texts [3], and we have embodied the model into a computer program called SNOWY [6,7]. When SNOWY starts reading a scientific paragraph its knowledge about the world consists only of a bare hierarchy of concepts. As SNOWY reads, new concepts and relations are learned by integrating them in the right place in the hierarchy. For instance, after reading the passage below with no knowledge about birds, SNOWY acquires the concepts *bird*, *egreet*, *heron*, *birds which live in swamps*, and *birds which live in forests*.

Birds are animals. Some birds, such as the egret and heron, live in swamps. Others live in forests.

There are several components in SNOWY, as illustrated in Figure 1. First, the sentences are parsed into a case structure using a parser based on the idea of syntactical usage of a word [4]. The output of the parser is passed to the Formation Phase, which consists of two components: Interpretation and formation. During interpretation the logical form for the sentence is constructed resolving prepositional attachment, anaphoric reference, etc. During formation properly said, the final knowledge representation structures are built from the logical form of the sentence. These knowledge representation structures are passed to the Recognition Phase, which checks if the concepts which have been formed exist in long-term memory (LTM). Those concepts which the Recognition Phase fails to recognize are passed to the Integration Phase which integrates them in LTM upon activating a Classifier Algorithm that indicates where and how they must be integrated in LTM. Finally, the Explanatory Phase tries to understand the explanatory links connecting sentences (e.g. Antibiotics work against infections. They kill the germs which cause them.)

Since SNOWY is reading novel texts and assimilating new concepts and relations, the system embodies a knowledge-limited model of comprehension, that is, one which does not involve schemas or frames to be instantiated as sentences are read. This differentiates SNOWY from knowledge-intensive models of comprehension based on scripts, plans or MOPs [2,12,13]. An essential aspect of comprehending in the face of limited knowledge is that of learning,

since successful comprehension involves the learning of new concepts and relations. There are several aspects of learning involved in SNOWY. The first one is that of acquiring classification hierarchies from texts [5]. For instance, from the sentences below

Not all animals which live in the sea are fish.
Some are mammals.

Snowy automatically constructs two subclasses of "animals," namely, "animals which live in the sea" and "fish," and two subclasses of "animals which live in the sea," namely, "sea fish" and "sea mammals." There are, of course, many other examples of sentences in which classification hierarchies must be learned if understanding is to occur.

A second aspect of learning involved in SNOWY is that of failure-driven learning. This occurs when SNOWY fails to recognize a relation or concept. For instance, if the system reads "antibiotics kill germs," knowing "antibiotics," but not "germs," the system will infer (from the concept "kill") that "germs" is an animate and will learn the relation underlying the sentence by integrating its conceptual representation in LTM.

However, in this paper we will present a third aspect of learning which is essential in understanding scientific prose, namely *explanation-driven learning*. In the next section, we introduce the concept. In section 3, we discuss the notions of analytical and empirical rules. Section 4 deals with the representation of concepts in SNOWY, and section 5 describes how analytical and empirical rules are activated. Finally, section 6 contains the conclusions.

2. Explanation-Driven Learning

Describing and explaining are two essential characteristics of scientific prose. Classification and the introduction of new concepts

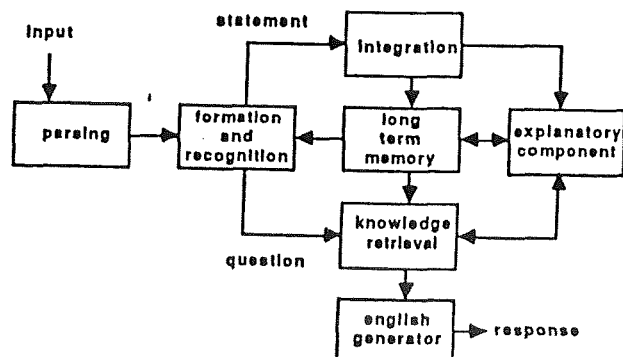


Figure 1. The Components of SNOWY

(*) This work was partially supported by NASA, Kennedy Space Center, under grant No. NAG-10-0058.

and relations are aspects of describing. Explaining is also an important aspect of scientific texts. This notion bears only a weak resemblance to the notion of explanation-based learning, which has recently become popular in research on learning [1,8]. Our notion of explanation was first presented in [3], and is based on the observation that there are two clearly distinct types of texts, which we have called explanatory and descriptive. An example of descriptive text is:

An alveolar air sac looks like a bunch of balloons.
The walls of each sac are very thin and moist. Alveoli are surrounded by a rich supply of capillaries.

In descriptive texts, concepts are described in terms of new or old concepts. In these texts, descriptive verbs and predicates predominate over action verbs. However, in explanatory texts sentences are explanations of relations introduced by previous sentences. Consider the following passages:

(1) Antibiotics work against infections. They kill the germs which cause them.

(2) Lions are very powerful. Few animals dare to bother them.

(3) Plants are dying. It is 20 degrees F outside.

In (1), the second sentence is an explanation of how antibiotics work against infections. In (2), the second sentence explains a consequence of lions being powerful. In (3), the second sentence is an explanation of why plants are dying. The understanding of these sentences requires the learning of the explanatory links which connect them. Without the acquisition of these links, one would not be able to answer the questions: How do antibiotics work against infections? Why do few animals dare to bother lions? or Why are plants dying? We have identified the following links between sentences. See [3] for a detailed description of these links.

1. Why do animate beings do things?
2. Why are animate beings in states?
3. What is the result or effect of states on those which suffer them?
4. Why are the objects of actions recipients of those actions?
5. What is the result or effect of physical actions on the objects of those actions?
6. What is the result or effect of actions on the instruments or recipients of those actions?
7. How are actions performed or how do they take place?
8. How much time do actions take?
9. How do animate beings behave or react to the actions performed by other animate beings towards them?
10. Why do certain things have certain properties?
11. What is the result or consequence of something having certain properties?
12. Why do certain things happen to be at certain places?
13. How small, large, etc. are things? The link consists of specifying fuzzy adjectives and adverbs. For example, "Plankton are tiny animals. Most of the plankton are one-thousandth of an inch."

3. Learning the Connections

There are two types of knowledge which allow the system to learn the links connecting sentences, which we have called analytical and empirical knowledge, analogous to the notions of analytical and empirical sentences. An analytical sentence is one whose truth conditions can be established by only having knowledge about the language. Classical examples of analytical sentences are "all bachelors are unmarried" and "all white horses are horses." The determination of the truth conditions of empirical sentences, on the other hand, requires knowledge about the world. For instance, the sentence "plants need light to live" requires knowledge about plants and light to establish its truth. We have used two types of rules to

encapsulate analytical knowledge and empirical knowledge, which we call analytical and empirical rules, respectively. In our opinion, Quine [10] has argued convincingly that there is not a dichotomy between analytic statements and empirical statements, but that the difference is one of degree. Yet, we consider that this distinction, although not dichotomous, is a relevant one, because we want to find out how much of the learning of new scientific concepts is based on analytical rules, and how much is based on empirical rules. The reader may find helpful to view analytical rules as very general common sense rules. Analytical rules are anchored in each primitive verbal concept. Some examples of analytical rules are:

- (1) If ingest(x,y) then enter(y,x)
- (2) If ingest(x,y) and located-on(z,y) then ingest(x,z)

The negation of rule (1) will result in a flagrant contradiction, one in which the meaning of "ingest" will be changed. Regarding (2), it is hard to imagine a world in which (2) does not hold. Let us consider the relation between analyticity and learning. Every time that an analytical rule is applied in order to find the connection between two sentences, a new piece of knowledge is learned. Consider the passage:

Antibiotics work against infections. They kill or slow down the germs which cause infections.

If after reading the first sentence we are asked "how do antibiotics work against infections?" we do not know the answer. Yet, after we read the second sentence, we can answer the question. This answer is based on an analytical rule which says:

If destroy(x,y) and cause(y,z) then work-against(x,z)

(where x is instantiated to antibiotics, y to germs, and z to infections). Note that the antecedent or premise of the rule is the explanation of how antibiotics work against infections. Then, the analytical rules are not only making possible understanding but also learning. In the above example, a new empirical fact has been learned, namely, that antibiotics are effective against infections because they destroy the germs which cause them. The analytical rule which has been used is of a general nature and, consequently, can be applied in many other different situations. Consider the passage:

During the San Francisco earthquake most houses burned. They were made of wood.

The connection can only be understood if we have an empirical rule which says that "if an object is made of wood, it burns." This notion of empirical rule is more manifest in the sentences:

Plants are dying. It is freezing outside.

It is generally accepted that it is a law of nature that if plants are submitted to temperatures below zero, they are affected negatively. The empirical nexuses connecting sentences are also found in the societal world, as exemplified in following sentences:

Lions are very powerful. Few animals dare to bother them.

The rule connecting these two sentences is not only true of lions, but also of politicians, countries, etc. In all these examples, the knowledge which connects the sentences can be considered to be domain specific common sense knowledge. Yet in other cases, very specialized knowledge may be needed to establish the connection. An entire scientific theory may mediate between the two sentences, as in the examples below:

There are worms in the meat. They grew spontaneously.
Cars are cheap. They are mass produced.

The first example invokes the theory of spontaneous generation to explain the appearance of worms in the meat. In the second example, a theory of mass production is needed for a thorough explanation of the link connecting the sentences.

Our present implementation includes a small set of empirical rules dealing with plants and animate beings in general. These rules are stored into the concepts in the hierarchy so that they can be inherited by children concepts.

4. Representation of Concepts

In this section we describe briefly the knowledge representation structures used in SNOWY. A detailed discussion can be found in [6]. Concepts are represented in LTM in frame-type structures, which are linked through *is-a* and *classes-of* slots to yield a hierarchical organization. Two kinds of representation structures may be part of LTM: *object-structures* and *action-structures*. An *object-structure* contains knowledge about physical or abstract objects and has slots called conceptual relations, or attributes. Two kinds of conceptual relations can be distinguished: those which describe objects and those which attribute actions to physical things. Thus, "centipedes are arthropods" and "all centipedes are slim and have many legs" are descriptive sentences, while "antibiotics kill germs" and "cells take in sugar" express actions performed by physical things. Descriptive attributes are represented following the frame notation. For instance, after reading the two sentences about centipedes, the *object-structure* built for *centipede*, will be the following:

```
(centipede
  (is-a (arthropod))
  (form (slim (q1 {all})))
  (has-body-part (leg (q1 {all})) (q2 {many})))
```

The slot *q1* contains the quantifier of the concept described by the *object-structure*, and the slot *q2* contains the quantifier of the second argument of the relation. Conceptual relations denoting actions, on the other hand, are represented through *action-structures*. For example, the relation underlying the sentence "antibiotics kill germs" is represented by the *action-structure*:

```
(@a00001
  (args (antibiotics) (germs))
  (pr (cause-to-die))
  (q1 {?})
  (q2 {?}))
```

The slot *args* contains all the arguments of the relation. The first concept in this slot is always the actor of the action verb and the second concept is the object. The slot *pr* contains the relation (*pr* stands for *primitive-relation*). The slots *q1* and *q2* contain the quantifiers of the first and second arguments, respectively. The term @a00001 is a symbol generated by the system, which stands for the name of the conceptual relation. *Action-structures* are linked to all the arguments of the relation. Thus, the complete representation of the sentence "antibiotics kill germs" consists of the following *object-structures* and *action-structure*:

```
(@a00001 (args (antibiotic) (germ)) (pr (cause-to-die))
  (q1 {?}) (q2 {?}))
(antibiotic (cause-to-die (germ ($more (@a00001))))
(germ (cause-to-die:by (antibiotic ($more (@a00001))))
```

Note how the *object-structures* of both, *antibiotic* and *germ*, point to the *action-structure* @a00001.

Let us now consider the sentences

Birds with long legs live in swamps. They have a long bill to extract food from under water.

This text introduces the concept "bird with long legs" for which a name is not given. However, a unique name in memory needs to be created for this concept so that additional knowledge about it can be learned by storing this knowledge under the same node, and so that this node can be linked to the other nodes in the hierarchy. Restrictive relative clauses and complex noun groups are represented in LTM via *object-structures* with a dummy name (a gensym), and their representation is characterized by the presence of a *characteristic-features* slot whose purpose is to identify the concept by describing the features that are characteristic of it. For example, when building the representation of the sentence, "Insulin is a hormone produced

by the pancreas," the concept *hormones produced by the pancreas*, a subconcept of the concept *hormones*, is created with the following representation:

```
(x1 (cf (is-a (hormone)) (make:by (pancreas (q2 {?}))))
```

Here, we use *cf* to denote *characteristic-features*. Within the *characteristic-features* slot there is always at least an *is-a* slot, which specifies an ancestor node of the concept being represented. In this example, the concept *x1* has been created as a subconcept of *hormones*, and further characterized by the fact that they are made by the pancreas. The newly-created concept is identified by the content of the *characteristic-features* slot and not by the dummy name "x1." Thus, if the question "what do hormones produced by the pancreas cause?" is asked, the system recognizes that the concept *hormones produced by the pancreas* was previously created and accesses the *object-structure* *x1*. (The slot *q2* contains "?" because it is unknown if every pancreas makes hormones.) Similarly, the representation of *bird with long legs* proceeds by creating the concept *long leg* first, and then the concept *bird with long leg*, as illustrated below.

```
x1 (cf (is-a (leg)) (length (long)))
x2 (cf (is-a (bird)) (has-body-part (x1 (q2 {?}))))
```

5. Activating the Rules

We now illustrate how the analytical and empirical rules are activated by following through two examples. Suppose the system reads the sentence

The human body recognizes the germs which enter it.

First, the sentence is parsed. The parser builds structures which identify the syntactical parts of the sentence. In our example, the parser output is the following:

```
g00008
(subj ((dfart the) (adj human) (noun body)) verb (recognizes)
  obj ((dfart the) (noun germs)) rela g00009)
g00009
(subj ((dfart the) (noun germs)) verb (enter) obj ((pron it)))
```

These structures become the input to the Formation Phase, which goes through the Interpretation and formation components in order to build the representation structures of the concepts and relations underlying the given sentence. In our example, the Formation Phase returns the following *object-structures* and *action-structures*:

```
(@a00009 (args (DMY00008) (human-body))
  (pr (enter)) (inside (human-body))
  (q1 {all}) (q2 {?}) (medium {?}) (how {?}) (source {?}))
(human-body (enter:by (DMY00008 ($more (@a00009))))
  (become-aware (DMY00008 ($more (@a00010))))
(DMY00008 (enter (human-body ($more (@a00009))))
  (cf (is-a (germ)) (enter (human-body (inside (human-body))))
  (become-aware:by (human-body ($more (@a00010))))
  (@a00010 (args (human-body) (DMY00008)) (pr (become-aware))
  (q1 {?}) (q2 {?}) (how {?})))
```

Figure 2. Representation structures built by the Formation Phase after reading the sentence "the human body recognizes the germs which enter it"

After completing the phases of parsing, formation and recognition, the Integration Phase inserts the newly built *object-structures* and *action-structures* into LTM. The integration of *object-structures* involves its classification among the concepts in LTM and, in some cases, they may trigger a reclassification of the concepts currently present in LTM. See [6] for a detailed discussion of this. When an *action-structure* is integrated in LTM, the integration component examines the definition of the primitive in this structure to determine which of its case slots have not been filled by the sentence just read. The unfilled case slots are placed in a list, a list of explanatory links, which is passed to the explanatory component. The explanatory

tory links generated for the sentence of our example are the following:

```
((medium enter (DMY00008 human-body) @a00009)
 (source enter (DMY00008 human-body) @a00009)
 (how enter (DMY00008 human-body) @a00009)
 (how become-aware (human-body DMY00008) @a00010))
```

Here, three links were generated for the action "germs enter the human body," and one link for "the human body recognizes DMY00008." The symbol *DMY00008* corresponds to the concept "germs which enter the human body." The symbols @a00009 and @a00010 are the names of the *action-structure* for which the corresponding link was generated. Once the explanatory component is activated, its first task is to determine if the sentence just read answers any of the links generated by previous sentences. These links are kept in the list ***links*, which at this point is empty, since we have read our first sentence. Next, the explanatory component examines the list of links just passed by the integration component, in order to determine if it is possible to answer any of them by using the analytical and empirical rules and the current knowledge in LTM (This situation occurs, for example, if the sentences "Lions are very powerful. Few animals dare to bother them" are read. In this case, the link "why do few animals dare to bother lions?" which would be generated from the second sentence, is answered by the first sentence). After doing this, the links that were passed to the explanatory component which are left unanswered are stored in the list ***links*. In our example, all four links shown above will be stored in ***links*.

This completes the processing of the first sentence. Let us now suppose that the next sentence read by the system is the following:

These germs are found in the food humans eat

After inserting the new concepts and relations in LTM, the integration component generates the links shown below, and activates the explanatory component.

```
((how ingest (human DMY00016) @a00017)
 (source ingest (human DMY00016) @a00017)
 (dest ingest (human DMY00016) @a00017))
```

These links correspond to the action "humans eat DMY00016," where the symbol *DMY00016* corresponds to the concept "food eaten by humans." The explanatory component now examines the representation structures built by the formation phase for the current sentence. In our example, the structures to be examined are the following:

```
(@a00017 (args (human) (DMY00016)) (pr (ingest))
 (q1 (?)) (q2 (all)) (how (?)))
(DMY00016 (ingest:by (human ($more (@a00017))))
 (cf (is-a (food)) (ingest:by (human)))
 (location-of (DMY00008)))
(human (ingest (DMY00016 ($more (@a00017))))
 (DMY00008 (location (DMY00016))))
```

Figure 3. Representation structures built by the Formation Phase after reading the sentence "These germs are found in the food humans eat"

For each relation introduced by the sentence, the analytical rules attached to its primitive are retrieved. In this case, the first relation examined is the action @a00017, so the rules attached to *ingest* are retrieved. One of these rules is the following:

```
((location ($x $y)) (ingest ($z $y))) (medium enter ($x $z))
```

The rule may be read as follows: "if *x* is located in *y*, and *z* ingests *y*, then *x* enters *z* through *y*." Once the analytical rules are retrieved, each link in ***links* is matched against the *then* part of the rules. At this point, after reading the two sentences above, the list ***links* contains:

```
((medium enter (DMY00008 human-body) @a00009)
 (source enter (DMY00008 human-body) @a00009)
 (how enter (DMY00008 human-body) @a00009)
 (how become-aware (human-body DMY00008) @a00010))
```

As we can see, the first link in ***links* matches the *then* part of the above rule, by instantiating variable *\$x* to *DMY00008* and variable *\$z* to *human-body*. Because there is a match, the explanatory component tries to verify the *if* part of the rule which, after instantiating the variables *\$x* and *\$z*, has the form:

```
((location (DMY00008 $y)) (ingest (human-body $y)))
```

To verify these two conditions, the explanatory component activates the knowledge retrieval component [7]. Each condition is passed to the knowledge retrieval component in the form of a question. Thus, the first condition becomes the question "where are germs which enter the human body found?" Since the knowledge the system has about germs is what has been specified in our two sentences, the answer to this question will be "germs which enter the human body are found in the food humans eat," thus instantiating the variable *\$y* to *DMY00016* (food humans eat). Then, the second condition in the *if* part becomes "(ingest (human-body DMY00016))," which is passed to the knowledge retrieval component as the question "does the human body ingest DMY00016?" Since the answer to this question is affirmative, the rule has fired successfully, providing an answer to the explanatory link "(medium enter (DMY00008 human-body) @a00009)." The explanatory component then proceeds to fill the *medium* case slot in the *action-structure* @a00009 with the concept *DMY00016* (food humans eat), and to remove the link "(medium enter (DMY00008 human-body) @a00009)" from ***links*. The rest of the links are examined in a similar way.

Let us now briefly consider a second example. Suppose the system reads the sentences:

Plants are dying. It is 20 degrees F outside.

After reading the first sentence, the explanatory link "(why negative-state(plant))" is generated. The answer to this link is obtained after reading the second sentence, by activating the empirical rule:

```
if less-than(temperature 32F) then negative-state(animate-being)
```

The rule encodes the fact that if the temperature is below 32 degrees, animate beings enter a negative-state (a semantic feature associated with such states as *death*, *infection*, etc.). The rule is stored under the concept *animate-being*, and inherited by the concept *plant* through the *is-a* hierarchy.

Using the Rules to Answer Questions

Let us consider the sentences

Antibiotics work against infections. They kill the germs that cause infections.

When the first sentence is read by the system, an *action-structure*, say @a1, will be built to represent the underlying relation, and one of the explanatory links generated will be "(how work-against (antibiotic infection))." This explanatory link will be answered by the second sentence because one of the analytical rules attached to the primitive *cause-to-die* is

```
(1) if cause-to-die(x y) and cause(y z) then (how (work-against(x z)))
```

which will fire successfully. Thus, the *how* case slot of the *action-structure* @a1 will be filled with the relation "cause-to-die (antibiotic x1)," where x1 is the concept "germs which cause infections." If the question "how do antibiotics work against infections" is now asked, a simple access to the structure @a1 in LTM will provide the answer. Suppose, on the other hand, that the system first reads the sentence:

Antibiotics kill the germs that cause infections

As before, once the knowledge in this sentence is integrated in LTM, the explanatory component retrieves the analytical rules attached to the primitives in the sentence, and fires them in an attempt to answer previously generated links. One of the rules retrieved is rule (1), above. However, contrary to our previous

example, this time after reading the sentence "antibiotics kill the germs that cause infections," the explanatory component will not add any new knowledge to LTM. The reason is that, since no sentences have been read before, there are no explanatory links to be answered. Therefore, questions like "how do antibiotics work against infections?" or "do antibiotics work against infections?" cannot be answered by the system. An obvious solution to this problem would be to fire all empirical and analytical rules in a forward fashion each time that a new sentence is integrated, and to add to LTM each piece of knowledge obtained. There are two problems with this solution. First, it takes a lot of processing time. A more serious problem, however, is that many irrelevant pieces of knowledge would be added to LTM. The solution we have adopted for this problem is the following. If the knowledge retrieval component does not find in memory the answer to a given question, it activates the explanatory component to try to infer the answer from memory. The explanatory component selects the analytical rules attached to the primitive of the verb in the question, and also the empirical rules, if any, attached to the main concepts in the question. This is what will happen if the question "do antibiotics work against infections?" is asked. The Explanatory Component gets the analytical rules attached to *work-against* and selects those whose *then* part have "work-against" in them (because the question has the primitive *work-against*). Then it tries to verify the truth of the *if* part of those rules. In our example, since rule (1) above is attached to *work-against*, as well as to *cause-to-die*, the Explanatory Component will instantiate this rule as:

If *cause-to-die*(antibiotics *y*) and *cause*(*y* Infection) then *work-against*(antibiotics Infection)

Then, in order to verify the truth of the *if* part of the rule, the Explanatory Component, in turn, activates the Knowledge Retrieval component. The Explanatory Component formulates the question "what do antibiotics *cause-to-die*?" The Knowledge Retrieval component will return a list containing the things killed by antibiotics. Then, the Explanatory Component instantiates the variable *y* to each of the elements in the list and, again, asks the Knowledge Retrieval component if any of those things cause infections. If the antecedent of the rule is found to be true, the answer to the question "do antibiotics work against infections" will be *yes*. The method we have explained allows us to obtain an analytical implication only when it is required to answer a question.

6. Related Research and Conclusions

There are striking differences between the type of learning described in this paper and explanation-based learning (EBL). In [8], the following four kinds of conditions need to be specified for EBL to occur.

- (1) *Necessary and sufficient* conditions expressing the concept to be learned.
- (2) A training example describing a positive example of the concept to be learned.
- (3) A *domain theory* consisting of a set of facts and rules which explain how training examples are instances of the concept to be learned.
- (4) An *operationality criterion* which specifies a set of predicates in which the concept to be learned needs to be reexpressed.

First of all, the algorithm explained in this paper does not learn concepts, but relations or connections between concepts. Secondly, and more importantly, the relations to be learned are not given to our system through the specification of necessary and sufficient conditions. The requirement that concepts to be learned be specified by defining them using necessary and sufficient conditions, constitutes, in our opinion, one of the most serious limitations in the current learning research. Our approach explores the learning of the connections which link the concepts in a web with no specific layers (see [11]).

In contrast to EBL, there is no training example in our method. Relations between concepts are learned as they are given in the text. Of course, the performance of our algorithm degrades if many sentences mediate between the introduction of a conceptualization

and its explanation, something that occurs in poorly written texts in which explanations are zigzagging, causing it to be difficult, even for good readers, to understand the text being read. These are issues we need to explore further and provide some quantifications for the concepts of "algorithm degradation" and "many mediating sentences."

There is a point of coincidence between requirement (3) in EBL, the domain theory, and our analytical and empirical rules. The main differences lie in the fact that, in EBL, the domain theory consists not only of rules, but also of facts. However, our domain theory consists only of rules. An important distinction is introduced in our domain theory by classifying the rules into analytical and empirical rules.

Finally, there is not a subset of predicates which constitutes the operationality criterion in our algorithm. A connection between two concepts is learned if predicates (not just a selected set of them) can be found in LTM which satisfy a rule or rules in the domain theory.

We have shown how a program can find and learn the explanatory connections existing between sentences. Our present implementation contains about forty analytical rules and ten empirical rules. Analytical rules are general in nature and are applicable to diverse domains. In fact, they are an essential component to the understanding of any domain. Empirical rules are domain dependent and do not play as essential a role as analytical rules.

Whether the distinction between analytical and empirical statements has any cognitive relevance has been the object of much debate in the philosophical and psychological literature. The study of Piaget and his coworkers [9] indicates that although human subjects do not "feel" a strong demarcation between these two types of sentences, yet they do sense that a relevant distinction is present in most cases. The preliminary feedback we have obtained from our program indicates that analytical and empirical knowledge are necessary elements in finding and learning the explanatory links which connect sentences in scientific discourse. From an engineering point of view, this distinction provides a relevant criterion for organizing the knowledge of a program in two distinct, although not dichotomous, categories. In our ongoing research, we are investigating the role that this distinction of analytical and empirical knowledge plays in knowledge-based problem solving.

References

- [1] DeJong, G. F. and Mooney, R. J. "Explanation-Based Learning: An Alternative View," *Machine Learning*, 1,2, 1986.
- [2] Dyer, M. G. *In-Depth Understanding*. MIT Press, Cambridge, MA, 1983.
- [3] Gomez, F. A Model of Comprehension of Elementary Scientific Texts, in *Proceedings of the Workshop on Theoretical Approaches to Natural Language Understanding*, Halifax, Nova Scotia, 70-81, 1985.
- [4] Gomez, F. WUP: A Parser Based on Word Usage, in *Proceedings of the 1988 IEEE Conference on Computers and Communication*, Scottsdale, Arizona, 445-449, 1988.
- [5] Gomez, F. Learning by Being Told: The Acquisition of Classification Hierarchies. Technical Report CS-TR-88-06, Department of Computer Science, University of Central Florida, Orlando, Florida, 1988.
- [6] Gomez, F. and Segami, C. The Recognition and Classification of Concepts in Understanding Scientific Texts. *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 1, 51-77, 1989.
- [7] Gomez, F. and Segami, C. Classification-Based Inferences in Retrieving Information from a Database of Scientific Facts. In *Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications*, Miami, Florida, 1989.

- [8] Mitchell, K. M., Keller, R. and Kedar-Cabelli, S. "Explanation-Based Generalization: A Unifying View," *Machine Learning* 1,1, 1988.
- [9] Piaget, J. (ed.) Les liaisons analytiques and synthetiques dans le comportement du sujet. *Etudes d'epistemologie genetique*. Paris, 1959.
- [10] Quine, V. W. *From a Logical Point of View*, Harvard University Press, 1953 (Revised 1963).
- [11] Quine, V. W. *Word and Object*, MIT Press, 1960.
- [12] Schank, R. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1982.
- [13] Wilensky, R. *Planning and Understanding*. Reading: Addison-Wesley, 1983.

APPENDIX

SHORT SAMPLE SESSION WITH SNOWY

In this sample session, lines preceded by ">>>" correspond to the user's input to SNOWY. Other lines, except for the remarks, correspond to SNOWY's response.

>>> the human body fights infections.
 >>> the human body kills the germs that cause infections.
 >>> how does the human body fight infections?

human-body work-against infection
 "how" <human-body cause-to-die <germ which cause infection>

REMARKS: After the integration of the second statement above, the Explanatory Component answers the link "how does the human body fight infections?," generated by the first statement, by firing the analytical rule "if x kills y and y causes z then this explains how x fights z." The explanation found is inserted in the appropriate *action-structure* in LTM, and the system is then able to answer the question "how does the human body fight infections?," as shown above.

>>> Insulin controls the level of sugar in the blood.
 >>> does insulin control sugar in the blood?

::: yes
 insulin control <sugar which location blood>
 because
 insulin control <level pertaining-to <sugar which location blood>

REMARKS: Here, the Knowledge Retrieval Component is unable to answer the question by accessing the current knowledge in LTM. Therefore, it activates the Explanatory Component, which fires the analytical rules attached to the primitive *control*. One of these rules is "if x controls y and y pertains to z, then x controls z." This rule fires successfully, and the system is able to answer the question, as shown above.

>>> the human body produces macrophages which kill bacteria.
 >>> does the human body kill bacteria?

::: yes
 human-body cause-to-die bacteria
 because
 human-body make <macrophage which cause-to-die bacteria>

REMARKS: This situation is similar to the previous one. Not being able to find an answer in LTM, the Knowledge Retrieval Component activates the Explanatory Component, which fires the rule "if x produces y and y kills z, then x kills z."

>>> bacteria cause infections to humans.
 >>> infections kill humans.
 >>> do bacteria kill humans?

::: yes
 bacteria cause-to-die human
 because
 infection cause-to-die human and
 bacteria cause infection "recipient" human

REMARKS: The analytic rule that allowed this question to be answered is "if x causes y to z and y kills z, then x kills z."

>>> germs which enter the human body cause diseases.
 >>> these germs are found in the food humans eat.
 >>> the human body kills these germs by producing antibodies.

REMARKS: The second statement provides the answer for the link "how do germs enter the human body?" generated by the first statement, and this new knowledge is inserted in LTM after integrating the second statement. Then, the system is able to answer the question below by accessing the LTM structures.

>>> how do germs enter the human body?
 germ enter human-body "medium" <food ingest:by human>

>>> does the human body fight diseases?

::: yes
 human-body work-against disease
 because
 human-body cause-to-die <germ which enter human-body> and
 <germ which enter human-body> cause disease

REMARK: The answer to this question is found by activating the analytical rule "if x destroys y and y causes z, then x work-against z."

>>> do antibodies kill germs?

::: yes
 human-body cause-to-die germ "how" <human-body make antibody>

REMARKS: In this case, the analytical rule applied is "if x kills y by producing z, then z kills y."

Appendix D

Knowledge acquisition from natural language for expert systems based on classification problem-solving methods

FERNANDO GOMEZ AND CARLOS SEGAMI

Department of Computer Science, University of Central Florida, Orlando, FL32816, USA

(Received 1 September 1989 and accepted in revised form 27 February 1990)

It is shown how certain kinds of domain independent expert systems based on classification problem-solving methods can be constructed directly from natural language descriptions by a human expert. The expert knowledge is not translated into production rules. Rather, it is mapped into conceptual structures which are integrated into long-term memory (LTM). The resulting system is one in which problem-solving, retrieval and memory organization are integrated processes. In other words, the same algorithm and knowledge representation structures are shared by these processes. As a result of this, the system can answer questions, solve problems or reorganize LTM.

1. Introduction

Can a simple expert system consisting of a hundred rules be designed from a natural language description by a human expert? Or can an expert system be updated from a natural language description of some of its components? In this paper, we present an integrated and domain independent system which builds classification-hierarchies from texts, retrieves information from them and solves problems.

We have been investigating the relationship between comprehension and problem-solving. Our investigation has been guided by the idea that problem-solving and comprehension are not two separate processes each one with its own knowledge representation structures, but that they spring from a common source and share the same knowledge representation structures. It is beyond the scope of this paper to discuss these issues in detail, rather we centre our discussion in providing evidence on how certain kinds of expert systems can be built from a natural language description by a human expert. We will show that the problem solving method used by the expert system to solve problems is the same as that used by the comprehender to keep memory organized and answer questions requiring chains of inferences.

We can roughly characterize the domain of the expert systems to which we have applied our method as consultation tasks. For instance, the tasks of building an expert system to select a programming language, or select a physician, etc. Some of these tasks are described in (Boose & Bradshaw, 1987). We can further characterize the scope of our system by the problem-solving method used to find solutions to a problem. The problem-solving method is that of classification

This research is supported by NASA-KSC Contract NAG-10-0058.

problem-solving (Gomez & Chandrasekaran, 1984; Clancey, 1985). Classification-problem solving consists of conceptualizing the domain of the problem-solver into a hierarchy of concepts and designing an algorithm which solves a problem by placing it in the hierarchy of concepts. In Gomez and Chandrasekaran (1984) it is stated:

Given these principles of organization of medical knowledge, the solution of a medical case becomes a problem of taxonomic classification. It is similar to the problem of placing, say, a specimen of maple in a hierarchy of botanical concepts. It consists of identifying its superordinate and subordinate concepts.

The method to recognize problem solutions by placing them in a hierarchy of concepts was that of using clusters of production rules organized under the concepts in the hierarchy, called *specialists* (Gomez & Chandrasekaran, 1984). In this paper, (a) we will show how to automatically build the classification hierarchies from natural language and (b) we will describe an algorithm which does not use the production rules to place the problem solutions in the hierarchy.

Among the knowledge acquisition systems which have been developed, ETS (Boose, 1984) and AQUINAS (Boose & Bradshaw, 1987) are the closer to our approach in the sense that these systems are domain independent and are based on building classification hierarchies for the solution of the problems. However, the differences are remarkable since we are not using elicitation techniques to build the classification hierarchies, but we are building them from natural language descriptions. Also, we are not translating the expert's knowledge into production rules, but we are using a general algorithm to produce solutions from the constructed hierarchies. Nano-KLAUS (Haas & Hendrix, 1980) was an earlier natural language acquisition system. However, this system is not targeted to the construction of expert systems, and it does not deal with the issues which are essential to our approach. The differences from other knowledge acquisition systems, MORE (Kahn, Nowlan & McDermott, 1985), MOLE (Eshelman, Ehret, McDermott & Tan, 1987) and SALT (Marcus, 1987), are more striking since these systems do not use classification problem-solving and presuppose domain knowledge.

This paper is organized as follows. In part 1, consisting of sections 2 and 3, we briefly explain some of the ideas on which our method is based, and in part 2, consisting of sections 4, 5, 6, 7, we describe how these ideas are applied to the construction of expert systems. Finally, we end with an appendix containing a brief session with SNOWY.

2. Background

For the last few years, we have developed a model of comprehension of elementary scientific texts or expository texts, and a program which embodies the model, called SNOWY, (Gomez, 1985; Gomez & Segami, 1989a,b). The comprehender of our model is a reader who is unfamiliar with most of the concepts which he/she is reading about. When SNOWY starts reading a scientific paragraph its domain knowledge consists only of a bare hierarchy of concepts. As SNOWY reads, new concepts are integrated in the right place in the hierarchy and new relationships about already known concepts are learned. For instance, after reading the passage

below with no knowledge of hormones and insulin, SNOWY acquires the concepts *hormones*, *insulin*, and *level of sugar in the blood*.

All hormones are chemicals produced by animals. Hormones control the activities of cells. Insulin is a hormone produced by the pancreas. Insulin controls the level of sugar in the blood by making cells take in sugar.

SNOWY does not only acquire those concepts, but it also builds a classification hierarchy. For instance, the concept *insulin* is classified as a subconcept of *hormone produced by the pancreas*; this, as a subconcept of *hormones*; this, in turn, as a subconcept of *chemical produced by animals*, and, finally, this last one as a subconcept of *chemicals*. In contrast to knowledge-intensive models of comprehension which understand by framing the input in rich pre-built structures, SNOWY understands by using *formation* rules which build conceptual structures from the logical form of sentences. These conceptual structures are then passed to a *recognizer* algorithm which checks if those concepts exist in long-term memory (LTM). Finally, those concepts which the recognizer algorithm fails to recognize are passed to an integration algorithm which decides where to integrate those concepts in LTM. We have called these three phases concept formation, concept recognition and concept integration.

Although SNOWY is a knowledge-lenient model of comprehension, this does not mean that SNOWY does not possess any knowledge when it starts reading a text. Besides the linguistic knowledge which allows SNOWY to parse and disambiguate sentences (Gomez, 1988), SNOWY consists of a set of categories organized in a hierarchy, a set of primitive relationships for action and descriptive verbs, which

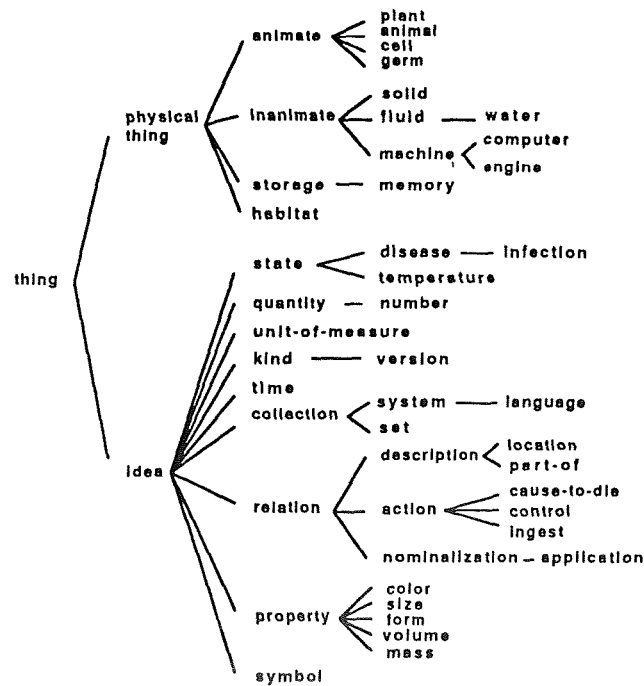


FIGURE 1. Partial content of the initial hierarchy of concepts in LTM.

are used to build the logical form of the sentence, and a set of formation rules which build the LTM conceptual structures from the logical form of the sentences. Figure 1 contains a sample of the *a priori* hierarchy, divided into physical things and ideas. The portion of the hierarchy which is shown in Figure 1 is strictly hierarchical. However, when the system begins to acquire new concepts, it builds a "tangled" hierarchy as described in the sections below. As new sentences are typed, the system builds the representation of the new concepts and relationships and inserts them into the hierarchy at the appropriate places. For example, for the sentence "germs are animates", the system adds the following to LTM:

```
(germ(is-a(animate)))
(animate(classes-of(germ)))
```

Before describing how to build these hierarchies, we need to explain briefly the knowledge representation structures used by our system. Three kinds of representation structures may be part of LTM: *object-structures*, *action-structures*, and *event-structures*.

An object-structure contains knowledge about physical or abstract objects and has slots called conceptual relationships or predicates. Two kinds of conceptual relationships can be distinguished: those which describe objects and those that attribute actions to physical things. This distinction corresponds to the distinction between descriptive and action verbs in natural language (Gomez, 1982). "Hormones are chemicals" and "all centipedes are slim and have many legs" are descriptive sentences, while "antibiotics kill germs" and "cells take in sugar" express actions performed by physical things. Descriptive attributes are also represented following the frame notation; for instance, the knowledge in the sentence about centipedes is represented as:

```
centipede
(form(slim(q1(all))))
(has-body-part(leg(q1(l(all)))(q2(many))))
```

The slot *q1* contains the quantifier of the concept described by the object-structure, and the slot *q2* contains the quantifier of the second argument of the relation. Conceptual relationships denoting actions are represented in the *object-structure* as:

relation (*a1*) (if the relation is monadic)

If the relationship takes two arguments or more it is represented as:

```
(relation (concept1 a1))
```

where *concept1* is the second argument of the relationship. If the same relationship is true of two or more concepts, as is the case of the relationship underlying the sentence "Penguins live in the sea and in the land", the relationship is represented in the *object-structure* as:

```
(relation (concept1 a1) (concept2 2a)...(concepti ai))
```

The terms *a1*, *a2*, ... *ai* stand for the names of the conceptual relationships. The structure which represents the conceptual relationship itself is called an *action-structure*. For example, the *action-structure* representing the conceptual relationship

in the sentence "antibiotics kill germs" looks like:

```
a1
(args (antibiotics) (germs))
(pr (cause-to-die))
(q1 (?))
(q2 (?))
```

The slot *args* contains the arguments of the relationship. If the relationship is monadic, the slot *args* will contain one concept. If the relationship is diadic (as in this case), the slot *args* contains two concepts, and so on. The first concept in the slot *args* is always the actor of the action verb and the second concept is the object. The slot *pr* contains the relationship. The slot *q1* contains the quantifier of the first argument, and the slot *q2* has the quantifier of the second argument of the relationship. If the relationship is triadic, there will be a *q3* slot for the quantifier of the third argument. The representation of "antibiotics kill germs" will be done by using the following *object-structures* and *action-structure*:

```
antibiotics      germs
(cause-to-die (germs a1))  (cause-to-die:by (antibiotics a1))

a1
(args (antibiotics) (germs))
(pr (cause-to-die))
(how (?))
(q1 (?))
(q2 (?))
```

Every concept which fills a case (actor, object, destination, source, etc.) in the sentence is indexed independently. In the above example, both the actor, *antibiotics*, and the object, *germs*, are indexed using an *object-structure*. Note how the concepts *germs* and *antibiotics* point to the same *action-structure*, *a1*, so that the question "what kills germs?" can be answered. The question mark in the slots *q1* and *q2* means that the sentence does not specify the quantifiers. If the sentence had said "all antibiotics kill germs", then the content of the slot *q1* would have been *all*. If the sentence had said "most antibiotics kill germs", the slot *q1* would contain *most*.

From a representation point of view, *event-structures are like action-structures*. However, they are used to represent relationships which are arguments of other relationships and, which can not be true if they are disconnected from the relationships in which they are embedded. For instance, it can not be concluded that "most people are happy" from the sentence "money causes most people to be

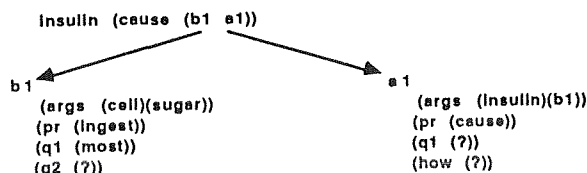


FIGURE 2. The representation of "insulin causes most cells to take in sugar".

happy". The sentence "insulin causes most cells to take in sugar", depicted in Figure 2, is represented by using three *object-structures* (insulin, cell, sugar), an *action-structure* (*a1*), and an *event-structure* (*b1*). (Gomez, 1986).

3. The detection and construction of classification hierarchies

The importance of explicit classification has been widely recognized and described by cognitive scientists working in reading. In explicit classification, the classes are clearly introduced and defined as in:

There are two kinds of mammals: terrestrial mammals and aquatic mammals. Aquatic mammals include the carnivorous and the omnivorous. The carnivorous includes the seals and walruses. The omnivorous includes the whales and etc.

In contrast to this type of classification, we will show that if a program is to acquire knowledge by reading texts it needs to classify in many instances in which classification is not apparent *prima facie*. Underlying classification is a more basic phenomenon than that which can be observed in other forms of classification, such as generalization-based classification or conceptual clustering. Furthermore, it is essential for constructing a problem solver from an English description of the task by a human expert, since the system will detect the classification hierarchies even when the expert does not see them. In the next sections, we will show that classification hierarchies underly two frequently used sentential structures: restrictive relative clauses and existentially quantified sentences.

3.1. LEARNING CONCEPTS DENOTED BY RESTRICTIVE RELATIVE CLAUSES

A first aspect of knowledge acquisition is posed by the learning of concepts denoted by noun groups and restrictive relative clauses, e.g. "birds with long legs", "hormones produced by the pancreas", "programming languages which run in PCs". The acquisition of these concepts involves their appropriate integration in the hierarchy, which in turn requires their classification by determining their parent and children nodes. For instance, consider the passage:

Birds with long legs live in swamps. They have a long bill to extract food from under water.

This text introduces the concept *bird with long legs* for which a name is not given. However, a unique name in memory needs to be created for that concept so that additional knowledge about this concept can be learned by storing this knowledge under the same node, and so that this node can be linked to the other nodes in the hierarchy. The acquisition of the concept *bird with long legs* entails its classification so that it can be integrated appropriately in the hierarchy of concepts. If the system fails to classify the concept, it will not be able to answer such questions as "are there birds with long bills?" or "do birds with long legs have feathers?"

Restrictive relative clauses and complex noun groups are represented in LTM via *object-structures* with a dummy name (a gensym), and their representation is characterized by the presence of a *characteristic-features* slot (written *cf*), whose purpose is to identify the concept by describing the features that are characteristic of the concept. For example, when building the representation of the sentence "insulin is a hormone produced by the pancreas," the concept *hormones produced by the*

pancreas, a subconcept of the concept *hormones*, is created with the following representation:

```
(x1 (cf (is-a (hormone)) (make:by (pancreas (q2 (?))))))
(hormones (classes-of (x1)))
```

Within the *characteristic-features* slot there is always at least an is-a slot, which indicates the parent node of the concept being learned. In this example, the concept *x1* has been created. *x1* stands for a subconcept of the concept *hormones*, as indicated by the is-a slot, and is further characterized by referring to those hormones made by the pancreas. The acquisition of the concept *x1* is completed by creating the slot *classes-of* in the concept *hormones*, and by filling it with *x1*. Had the slot *classes-of* existed in the concept *hormones*, then *x1* would be added to it. The newly-created concept is identified by the content of the *characteristic-features* slot and not by the dummy name *x1*. Thus, if the question "what do hormones produced by the pancreas cause?" is asked, the system recognizes that the concept *hormones produced by the pancreas* was previously created and accesses the *object-structure* *x1*. (The slot *q2* contains "?" because it is unknown if every pancreas makes hormones).

The acquisition of concepts described by complex noun groups is done in a similar manner. The representation of *bird with long legs* proceeds by creating the concept *long leg* first, and then the concept *bird with long legs*. Both concepts are represented in Figure 3. This definition creates the concept *x1*, a subconcept of the concept *leg*. The concept *bird with long legs* is represented by *x2*. The *characteristic-features* slot in *x2* says that *x2* is a subconcept of *bird* characterized by having long legs. Because the noun group does not specify how many long legs *x1* has, the quantifier slot *q2*, whose scope is *x1*, has a question mark in it. If the system has knowledge about birds, it fills this slot; otherwise, it will remain with a question mark until the system finds out more about birds.

3.2. FORMATION OR LEARNING RULES

The mechanism we have described for acquiring concepts denoted by restrictive relative clauses and noun groups is based on *formation or learning rules*. These are very general rules anchored in the verbal primitives, and in some instances in lexical items. In most cases these rules use knowledge in LTM and in the structure built by the parser.

The formation rules which form the concepts *x1* and *x2*, which have been explained above, are part of the formation of noun groups. For instance, the antecedent of the rule which is fired in the example above is:

The head noun has the feature *animate* and is followed by the preposition "with", and the object of the preposition has the feature *body-part*.

<i>x1</i>	<i>x2</i>
(cf(is-a(leg))(size(long)))	(cf(is-a(bird))(body-part(x1 (q2(?)))))
<i>leg</i>	<i>bird</i>
classes-of(<i>x1</i>)	classes-of(<i>x2</i>)

FIGURE 3. The representation of "bird with long legs".

Since the object of the preposition is also a noun group, the formation rules for the noun group are activated recursively to form the concept *long leg*. In this case, the antecedent of the rule activated is:

The head noun has the feature body-part and its modifier has the feature size.

The latter rule will form the concept *x1* and will pass it to the former rule, which will form the concept *x2*.

This has been a very simple example to illustrate basic formation rules. In the next section, we explain how formation or learning rules are used to learn more complex classification hierarchies.

3.3 THE ACQUISITION OF CLASSIFICATION HIERARCHIES

As we have seen, a classification hierarchy may be indicated explicitly, as in the sentence "antibiotics are chemicals" or it may be stated in a more subtle way as in the phrase "hormones produced by the pancreas". There is yet a more complex way of expressing a complex hierarchy of concepts like those illustrated in the passage below:

(1) Not all animals which live in the sea are fish. Some are mammals.

These two sentences introduce three concepts: *animals which live in the sea*, *sea fish* and *sea mammal*. The last two concepts are subconcepts of the first one.

The learning of these concepts and the hierarchy which they form is done by means of a learning rule attached to *not all* and by one of the learning rules anchored in *are*. The first sentence is parsed into:

```
(subj (not all animals) rela (g00001) verb (are) pred (fish))
g00001
(subj (animals) verb (live) prep (in the sea))
```

The learning rule which builds the representation for restrictive relative clauses is activated. This rule activates the formation rule for the verb *live*. The structure built is:

```
x1                                animal
cf(is-a(animal) (habitat (sea)))  classes-of (x1)
```

The name of the concept which has been created replaces the concept modified by the relative clause in the main sentence, resulting in:

```
(subj (not all x1) verb (are) pred (fish))
```

The formation of the main sentence proceeds by activating the learning rules for *are*. The following rule fires:

- If the subject is under the scope of an existential quantifier or the negation of an universal quantifier then
- create a concept with a dummy name, say *x2*, and with a characteristic-features slot containing the slots *is-a (subj)* and *is-a (pred)*
 - create another concept with a dummy name, say *x3*, and with a characteristic-features slot containing the slots *is-a (subj)* and *is-a (?)*.

The terms *pred* and *subj* in the rules are variables containing the subject and the predicate of the sentence, respectively. This rule when applied to (1) gives:

x2	x3
(cf (is-a (x1) (fish)))	(cf (is-a (x1) (?)))

The application of the rule for restrictive relative clauses and the one for *are* has resulted in the creation of three concepts: the concept *x1*, which corresponds to *sea animal*, the concept *x2*, which corresponds to *sea fish*, and *x3*, which has been only partially specified.

Adverbs such as *almost*, *only*, and the negation of universal quantifiers, e.g. *not all*, introduce links which connect the sentence where they appear to the next sentence. In order to solve these links, these lexical terms have attached to them formation rules. These rules are activated as part of the formation of the sentence which immediately follows the one which starts with *not all*, etc. These formation rules resolve the anaphoric reference of *some* in the second sentence and fill in the question mark in the is-a slot of the concept *x3* above. The concepts and the hierarchy formed for the passage *Not all animals which live in the sea are fish. Some are mammals* are found in Figure 4. The formation rules for *not all* do not only deal with this example, but also with examples such as:

Not all animals which live in the sea are fish. Whales are mammals.

In this case, the formation rule checks LTM to see if whales live in the sea. If so, it will replace the question mark in the concept *x3* above with *mammal*, and create *whale* as a subconcept of *x3*. However, even if there is no knowledge about *whales* in LTM, the system infers from the structure of the sentence that *whales* are sea animals.

3.4. THE ACQUISITION OF CONCEPTS DENOTED BY EXISTENTIALLY QUANTIFIED SENTENCES

The acquisition of concepts introduced by sentences starting with an existential quantifier requires the construction of a classification hierarchy in the same manner as we have done for relative clauses. The sentence "some mammals live in the sea" introduces the concept *mammal which live in the sea*. However, the representation in Figure 5 is inadequate for the job. The problem with this representation becomes obvious if the next sentence is "these mammals are intelligent". Since the structure above has not created the subconcept *mammal which live the sea* there is no node in

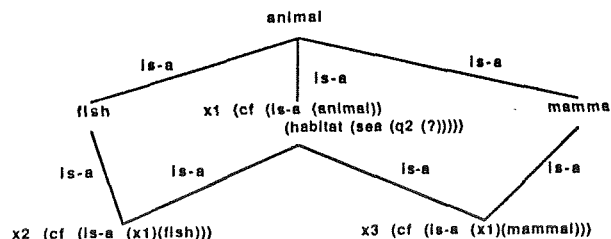


FIGURE 4. The hierarchy constructed for "Not all animals which live in the sea are fish. Some are mammals".

```

mammal
habitat (sea a1)

sea
habitat: by (mammal a1)

a1
args(mammal,sea)
pr (habitat)
q1 (some)
q2 (?)

```

FIGURE 5. Inadequate representation of "some mammals live in the sea".

the hierarchy to integrate the knowledge that they are intelligent. The representation of the sentence "some mammals live in the sea" is given below:

$x1$ (cf(is-a(mammal)) (habitat (sea(q2(?))))))

$x1$ represents the concept *animals which live in the sea*. The concept *mammal* will contain the slot "classes-of ($x1$)". The knowledge expressed by the sentence "these mammals are intelligent" can now be integrated under the concept $x1$.

The role of classification is one of the most pervasive aspects in the structure of expository prose. As a final example consider the text below:

Some birds are unusual because they cannot fly. The emu, the ree, and the ostrich are such birds. The emu is an Australian bird. The ree lives in South America. The ostrich lives in Africa and runs very fast.

Some birds are unusual because they form a partnership with an animal. The oxpecker eats insect from the rhinoceros. The honey guide will lead a badger to a bees' hive.

```

-----
-----
-----
-----
-----
-----
-----
-----

```

The authors found this text in a 6th grade comprehension exercise to teach the notion of classification to children. The children were asked to write the names of birds on the dotted lines which follow the text. This text illustrates clearly that only through classification can the concepts and relationships in the text be learned.

4. The application of these ideas to the automatic construction of expert systems

We will show in this section how these ideas can be applied to the construction of an expert system. The task which we will study is that of building a consultation system. We will use an example similar to that discussed in Boose and Bradshaw, 1987, the selection of a programming language.

The expert is instructed to present his/her subject in a top down fashion describing first the most general concepts and then refining them until reaching the most concrete concepts which will form the tip nodes in the hierarchy. The paragraph below is just one way of how this description may be accomplished. Many

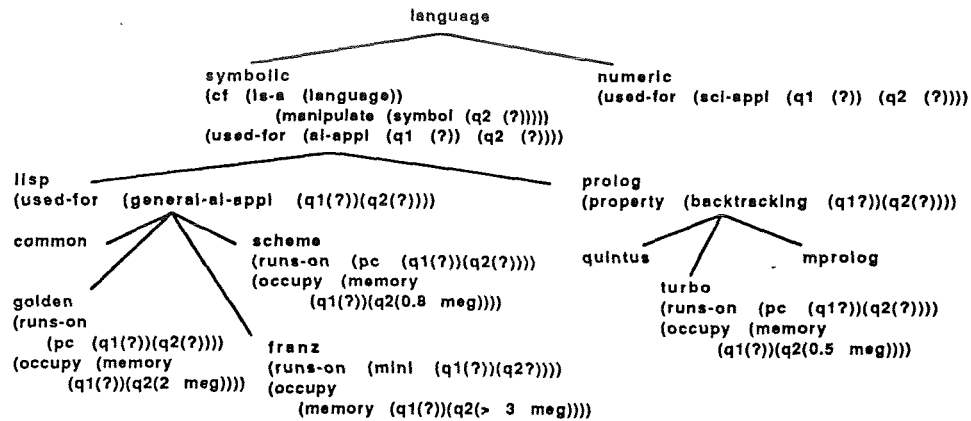


FIGURE 6. Hierarchy for the Text on Programming Languages.

others are possible. After reading this paragraph, SNOWY builds the hierarchy of concepts shown in Figure 6.

There are several kinds of programming languages. Programming languages which manipulate symbols, called symbolic languages, are used for AI applications, while numeric languages are used for scientific applications. There are two main symbolic languages: Lisp and Prolog. Lisp is used for general AI applications, whereas Prolog is used when backtracking is needed. Common Lisp, Franz, Scheme and Golden Hill are Lisp languages. Franz runs in minicomputers and takes more than 3 megabytes of memory. Scheme is a dialect of Lisp for PCs. It takes 0.8 megabytes of memory. Golden-Hill is also a version of Lisp for microcomputers, but it requires 2 megabytes of memory. There are several versions of Prolog, such as Quintus, Turbo and Mprolog. Of these, Turbo runs on PC's, and requires only 0.5 megabytes of memory. Numeric languages. . . .

The system proceeds by parsing the sentences. In the current implementation, the system can parse any sentence containing unknown nouns and adjectives, but verbs must be known to the parser for a successful parse to occur. If the parser fails then the system asks the user for a definition of those words which are unknown to the system. First of all, the system asks the user if the word in question is a verb (the user can check that by consulting a dictionary). If the word is a verb the system obtains the syntactical usage of the verb by activating an interface which allows users without a formal knowledge of English grammar to add new verbs to the parser. The system can make sense of many sentences containing words unknown to it. We postpone the discussion of how this is done until the next section. We briefly explain some of the formation rules used to construct the hierarchy in Figure 6. The first sentence does not create any structure. From the second sentence, the system creates the concepts *symbolic language* and *numeric language*, according to the techniques which we have explained for the formation of concepts denoted by restrictive clauses and noun groups. The *characteristic-features* slot of the concept *symbolic language* contains the conceptual relationship "manipulate symbols". Note that from this moment on, the meaning of *symbolic language* is a programming language which manipulates symbols. The formation of "symbolic languages are

used for AI applications", is integrated by creating the descriptive relationship "used-for" (ai-application)" under the concept *symbolic language*. The next sentence creates two subconcepts of *symbolic language*: Lisp and Prolog. The next sentence differentiates the two languages. The formation rule which builds the LTM representation of "Prolog is used when backtracking is needed" constructs the following structure:

```
Prolog
  (property(backtracking))
```

This is a general rule which forms all logical forms which fall under the schema:

```
< somebody > < use > < obj1 > when/if < somebody > < need > < obj2 >
```

into the structure:

```
obj1(property(obj2))
```

The rule is using the inference that if something, say x , is used when something else, say y , is needed then y is a property of x . This is necessary if requests such as "I need a symbolic language with backtracking" are to be answered by the consultation system.

5. Defining unknown words

One of the critical issues in the design of the system is how many concepts in the user's input need to be previously known by the system in order for it to understand the text and solve problems. If the knowledge acquisition system is going to be domain independent, then the system can not have *a priori* knowledge about concepts which are domain specific. For instance, the concept of "backtracking" in the sample text can not be part of the *a priori* concepts. However, if the knowledge acquisition system is going to be domain-dependent, then one can provide it with a rich set of domain dependent concepts. Since our goal is to build a domain independent knowledge acquisition system, the problem of unknown concepts becomes critical. The sample text can be processed by the present implementation without knowing anything about the following concepts: "programming languages", "symbols", "AI", "numeric languages", "backtracking", "Lisp languages", "megabyte" and "Lisp", "Prolog", etc. If one asks the system "what is backtracking", this will reply by saying "I do not know what it is, but I know that it is a property of Prolog". The system's reply to a question is "I do not know" if the concept in the question is not linked through the *is-a* relationship to the concepts in the *a priori* hierarchy. For classification problem solving, the system does not need to have a deep understanding of what backtracking is. In any case, the decision of determining if a concept should be fully known to the system rests on the user.

In some cases, however, it is essential to know the meaning of some terms, because without knowing them the meaning of verbs, prepositions and attachment of prepositions can not be determined. If one does not know the meaning of "evolution" in the sentence "Peter read a book on evolution in 5 days", the attachment of "in 5 days" can not be decided. Let us discuss these issues in the context of acquiring the other concepts in the text. The sentence in which the

concept "megabyte" appears for the first time is "Franz ... takes 3 megabytes of memory". Let us assume that the system knows that memory is something where information is stored. The problem here is that when the system is going to form the final knowledge representation from the logical form of the sentence, it cannot determine the meaning of "of" because it does not know the meaning of "megabyte". In these cases, the system displays the possible meanings of "megabyte" from the context and asks the user to select the most appropriate one. In this case, the system selects these meanings from the formation rules which are used to determine the meaning of "of". (The actual display for this case is illustrated in the appendix.) For instance, the system will display:

Is megabyte a kind of memory?
 Is megabyte made of memory?
 Is megabyte a unit of measure of memory?

.....

In other cases, the user may decide to define a concept which is unknown to the system by using one of the three methods explained briefly below.

- *Classification*: The best way to define a word is by providing the upper-concept of the concept denoted by the word to be defined. For instance, the word AI can be defined by saying "AI is a computer science". An inappropriate definition will be to say something like "AI tries to make computers intelligent", since this does not place AI in the hierarchy of concepts which forms the basis of SNOWY's understanding process. Likewise, the definition which the system is expecting of *backtracking* is not a long and intricate one, which probably will be irrelevant to the expert system being built, but just something like "backtracking is a programming technique" which places *backtracking* in the hierarchy of concepts. We are extending the *a priori* hierarchy of categories to include a large set of concepts so that it will be easy to place new concepts in the hierarchy by classifying them.

- *Part-of definitions*: Some words may be defined by using a *part-of* relationship. For instance, the word *cpu* or *computer memory* can be defined by saying they are part of a computer.

- *Functional definitions*: We are studying short functional definitions of concept from which the system can infer the *part-of* relationship plus some other information. For instance, the concept *computer memory* above can be defined as: "memory is something to store information into a computer". From this, the system can infer that *computer memory* is a physical thing, part of a computer and is used to store information. The reason why this method of definition has not been incorporated in the system, yet, is because it opens the door for a user to give a definition too complex to be understood.

6. Problem-solving

After SNOWY has read the paragraph of section 4, a user interested in a consultation about programming languages may, at this point, run a consultation test. For example, the user may say "I am interested in a language that runs on PC's

and is used for ai applications". Given this inquiry, the task of the system is to find the languages that have the features "runs on PC's" and "used-for ai applications". The class of languages that have these features is represented as:

$x1$ (cf(is-a(language)) (runs-on(pc)) (used for (ai-appl)))

If this concept were present in LTM, the question would be answered by simply retrieving its children. Since this is not the case, one way to find the answer to the question is to *classify* the concept $x1$ ("languages that run on PC's and are used for ai") within the current hierarchy of concepts. This task is done by a *classifier algorithm*, which determines where in the current hierarchy a new concept, such as $x1$, would be located if it existed in LTM, that is, the *classifier* determines which of the concepts currently in LTM are the parents of $x1$ and which ones are its children. We briefly illustrate the steps taken by the *classifier*, using the concept $x1$ as an example. See Gomez and Segami (1989a) for a detailed description of the algorithms in this section and a comparison with other classifier algorithms. Given the representation structure of $x1$, the concept within its *is-a* slot determines what part of LTM needs to be considered by the *classifier*. In this case, only the concept *language* or its children can be the parents or the children of $x1$. The *classifier* performs a traversal of selected nodes under the concept *language*. The concept in each visited node is compared with the concept $x1$. Depending on whether one is a subclass of the other, or there is no hierarchical relationship between them, the *classifier* decides which node to visit next. The actual comparison of concepts is performed by the *compare-classes* algorithm, which is based on the following idea. A concept $x2$ is a subclass of a concept $x1$, if each characteristic-feature of $x1$ is true of each entity in the class $x2$. In our example, concept $x1$ is first compared with the concept *symbolic languages*. No hierarchical relationship is found between these two concepts. However, one of the *characteristic-features* of $x1$, "used-for ai applications", is found to be true of all the descendants of the concept *symbolic languages*, so that, when the nodes under *symbolic languages* are examined, only one more feature needs to be verified: "runs on PC's". After doing this, it is found that *scheme*, *golden*, and *Turbo* run on PC's. Then, the *classifier* returns the list "[language) (scheme golden turbo)]" to indicate that the concept "languages that run on PC's and are used for ai" is a child of the concept *language* and that it has the children *scheme*, *golden*, and *Turbo*. The system, then, would reply to the user as follows: "the following languages run on PC's and are used for ai: scheme, golden, and turbo". If the user decides to narrow down his/her choices, he/she may either specify additional features, or formulate questions. For example, the following interaction may now take place:

user: the language must run in a computer with less than 1 megabyte of memory

system: scheme and turbo require less than 1 megabyte of memory

user: what are the differences between scheme and turbo?

system: scheme is a lisp dialect used for general ai applications while turbo is a prolog dialect used for backtracking

The first sentence typed by the user simply adds one more requirement to the desired programming language. Since we have an initial set of solutions, the new requirement needs to be verified for the members of this set only. This reduces the

possible solutions to two. The second sentence typed by the user is a question, which requires that a comparison be made between the features of *Scheme* and *Turbo*. This is accomplished by traversing the hierarchy upwards, from the nodes corresponding to *Scheme* and *Turbo*. The traversal stops when the two paths cross each other.

The *classifier* is the component that allows SNOWY to maintain a correct organization of its memory. Each new statement entered by the expert may cause the insertion of a new concept in the hierarchy, or it may trigger a reorganization of the concepts in LTM. Let us suppose, for example, that the expert continues entering information to the system, by typing the following:

Some languages are procedural. All procedural symbolic languages are widely available.

The sentence "some languages are procedural" is integrated as explained in section 3.4. SNOWY's long-term memory configuration for the two sentences is depicted in Figure 7. If the expert now types the statement: "Lisp is a procedural language", the concepts in LTM will be reorganized as shown in Figure 8. Thus, in an inquiry of the form "a language that runs on PC's is used for ai applications, and is widely available", only the Lisp dialects will be considered to be possible solutions.

As one can see the problem-solving method which we have illustrated uses only categorical knowledge (Szolovits & Pauker, 1978) and not probabilistic knowledge. It is our belief that probabilistic reasoning plays a minor role in most problem-solving situations if memory organization and indexing form part of the problem-solving method. In any case, certain aspects of probabilistic reasoning need to be included in our system. Experts use sentences like this "Lisp is easier to learn than Pascal which in turn is easier to learn than Cobol". In cases like this, we need to ask

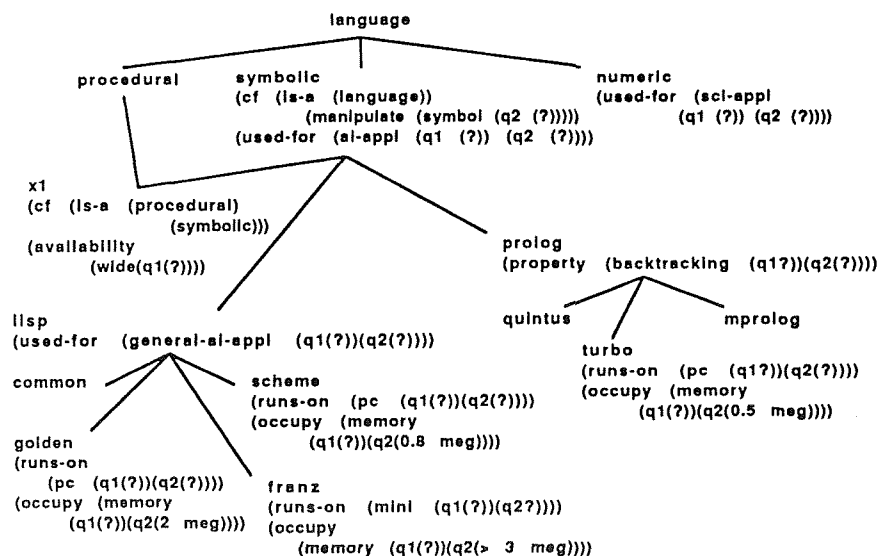


FIGURE 7. Organization of LTM after processing the passage "Some languages are procedural. All procedural symbolic languages are widely available".

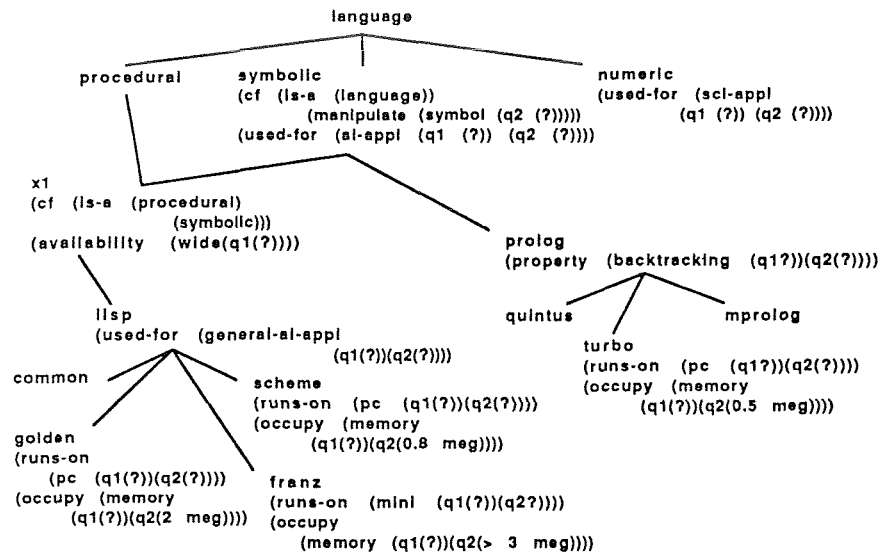


FIGURE 8. Reorganization of the concepts in LTM after processing the sentence "Lisp is a procedural language."

the expert to rate Lisp, Pascal and Cobol with respect to the predicate *learn*. There are many ways in which this rating can take place and AQUINAS offers several ways which we plan to adopt in our system. Let us consider for the sake of the discussion that the expert produces the following rating: (Lisp (learn ·9)) (Pascal (learn ·7)) (Cobol (learn ·4)). Then, the phrase "a symbolic language that is easy to learn" in the user's request "I am interested in a symbolic language which runs in a PC and is easy to learn", will be translated into a numeric value which may be greater than 5 according to the rating provided by the user for that predicate.

7. Discussion

In this paper, we have shown how a consultation system can be built from a natural language description by a human expert. It is our belief that the ideas and techniques described here can be used for the construction of expert systems for tasks which can be modelled using classification problem-solving. These tasks do not have to deal with consultation systems. They may be about diagnosis, design, etc. The relevance of the ideas presented in this paper go beyond the knowledge acquisition area, since we have introduced a problem-solver which has striking differences from existing rule-based systems. In our system, organization and reorganization of memory, comprehension and problem solving are integrated processes. As a result, two important aspects of expert system technology become almost trivial: the updating of the knowledge base and interacting with the expert system. However, our approach is not without limitations. Let us first discuss the theoretical limitations. Although we believe that a large class of problems and domains can be handled using our system, there are many domains which are beyond reach of our method. The reason for that is that there are many aspects of

the real world which can not be represented using the classification hierarchies we have presented in this paper. For instance, the functioning of complex systems such as the circulatory system, or the respiratory system can not be adequately grasped using our present representation techniques. We are working in knowledge representation structures which will represent these systems from a point of view which will be adequate for comprehension and problem-solving (Gomez, 1990).

Although the capabilities of the system have grown considerably since the submission of this paper, there are limitations to our present implementation. First of all, no expert has yet used our program to build a consultation system. All the consultation examples have been entered by graduate students or by the implementers of the system. Our system is a prototype which we can "demo" at any time, but that has not been used by any real user. This is clearly in contrast with AQUINAS which, according to the authors, has been used to build a variety of consultation systems. We are going to start to experiment with some real users to assess the real world capabilities of our system. Since SNOWY is able to acquire new concepts and new words, it can augment its lexicon by interacting with a user. The parser (Gomez, 1988) being used by SNOWY is a modular algorithm based on the notion of *syntactical usage of a word*, which allows a naive user without knowledge of English grammar to increase its syntactical coverage. We are writing an interface to our parser which will allow a user to add new verbs not presently in the parser. Yet, we are aware that there is a qualitative jump from testing a prototype by people who have certain familiarity with it to a real world system used by naive users.

We have presented a system and distinguished it from existing knowledge acquisition systems. Yet, our approach can be integrated with present knowledge acquisition tools and vice versa, techniques in other systems can be incorporated in our system. We think that, in future developments of our system for real users, the elicitation techniques developed for ETS and AQUINAS can be very useful. In particular, we are thinking that in those cases in which the English of certain users becomes too hard to handle, SNOWY may fall back on elicitation techniques. Likewise, domain dependent knowledge acquisition tools can benefit from a system which is able to build classification hierarchies from natural language text.

References

- BOOSE, J. (1984). Personal construct theory and the transfer of human expertise. *Proceedings of the National Conference on Artificial Intelligence, Austin, Texas.*
- BOOSE, J. & BRADSHAW, J. (1987). Expertise transfer and complex problems: using AQUINAS as a knowledge acquisition workbench for expert systems. *International Journal of Man-Machine Studies*, **26**, 3-28.
- CHARNIAK, E. & McDERMOTT, D. (1984). *Introduction to Artificial Intelligence*. Reading, MA: Addison-Wesley.
- CLANCEY, W. J. (1985). Heuristic classification, *Artificial Intelligence* **27**, 289-350.
- ESHELMAN, L., EHRET, D., McDERMOTT, J. & TAN, M. (1987). MOLE: a tenacious knowledge acquisition tool. *International Journal of Man-Machine Studies*, **26**, 41-54.
- GOMEZ, F. (1982). Towards a theory of comprehension of declarative texts. In *Proceedings of the 20th Meeting of the Association of Computational Linguistics, Toronto, Canada*, 36-43.

- GOMEZ, F. (1985). A model of comprehension of elementary scientific texts. In *Proceedings of the Workshop on Theoretical Approaches to Natural Language Understanding, Halifax Nova Scotia*, 70–81.
- GOMEZ, F. (1986). *Knowledge Representation for Understanding Expository Texts*. Technical Report CS-TR-86, Department of Computer Science, University of Central Florida, Orlando, Florida.
- GOMEZ, F. (1988). WUP: A parser based on word usage, in *Proceedings of the 1988 IEEE Conference on Computers and Communication, Scottsdale, Arizona*, 445–449.
- GOMEZ, F. & CHANDRASEKARAN, B. (1984). Knowledge organization, and distribution for medical diagnosis. In W. CLANCEY & E. SHORTLIFFE, Eds. *Readings in Medical Artificial Intelligence*. Reading, MA: Addison-Wesley.
- GOMEZ, F. & SEGAMI, C. (1989a). The recognition and classification of concepts in understanding scientific texts. *Journal of Experimental and Theoretical Artificial Intelligence*, **1**, 51–77.
- GOMEZ, F. & SEGAMI, C. (1989b). Classification-based inferences in retrieving information from a database of scientific facts. In *Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications, Miami, Florida*.
- GOMEZ, F. (1990). *Representing Biological and Physical Systems as Temporal Event Hierarchies*. Technical Report CS-TR-90-02, Department of Computer Science, University of Central Florida, Orlando, Florida.
- HAAS, N. & HENDRIX, G. (1980). An approach to acquiring and applying knowledge. *Proceedings of the National Conference on Artificial Intelligence, Stanford, CA*, 235–239.
- KAHN, G., NOWLAN, S. & McDERMOTT, J. (1985). MORE: an intelligent knowledge acquisition tool. *Proceedings of Ninth International Conference on Artificial Intelligence, Los Angeles, California*.
- MARCUS, M. (1987). Taking backtracking with a grain of SALT. *International Journal of Man-Machine Studies*, **26**, 383–398.
- SZOLOVITS, P. & PAUKER, S. (1987). Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence*, **11**, 115–144.

Appendix I

SAMPLE SESSION WITH SNOWY

NOTES: (1) SNOWY's knowledge before reading the sentences below, is described by the hierarchy of Figure 1.

(2) Lines preceded by ">>>" correspond to the user's input to SNOWY. Except for the remarks, other lines correspond to SNOWY's response.

(3) Symbols of the form *dmyxx* denote *object-structures* having a *characteristic-features* slot.

```
>>> what is a language?
:::language is-a system

>>> what languages are there?
:::I don't know
```

REMARKS: As indicated in the hierarchy of Figure 1, initially SNOWY knows that the concept *language* is a subconcept of *system*. Since SNOWY does not have any other knowledge about *language*, the question "what is a language?" is answered "language is-a system", and the response is "I don't know" when the question is "what languages are there?"

>> > there are two kinds of programming languages.
 >> > programming languages which manipulate symbols, called symbolic languages, are used for ai applications, while numeric languages are used for scientific applications.
 >> > lisp and prolog are symbolic languages.
 >> > fortran is a numeric language.
 >> > lisp is used for general ai applications.
 >> > prolog is used when backtracking is needed.
 >> > franz and scheme are lisp languages.

REMARKS: The user has typed several sentences which SNOWY has parsed, formed and integrated in LTM. Concepts such as *kind*, *symbol*, plus all the verbal primitives are part of the initial hierarchy of concepts. Other concepts such as *programming languages which manipulate symbols*, *scientific application*, *backtracking*, etc., are formed and inserted in LTM.

>> > franz runs on minicomputers and takes 3 megabytes of memory.
 :::unable to determine meaning of verb: *runs-on*
 :::because concept *minicomputer* is unknown
 :::please indicate which of the following is the most appropriate statement:
 :::(1) minicomputer is a kind of machine
 :::(2) minicomputer is a kind of computer
 >> > 2.
 :::integrating *minicomputer* as a subconcept of *computer*
 :::unable to form prepositional phrase: *3 megabytes of memory*
 :::because concept *megabyte* is unknown
 :::please indicate which of the following is the most appropriate statement:
 :::(1) megabyte is a kind of memory
 :::megabyte is made of memory
 :::megabyte is a unit of measure of memory
 :::megabyte is a part of memory
 >> > 3.
 :::integrating *megabyte* as a subconcept of *unit-of-measure*

REMARKS: When SNOWY reads this sentence, it finds that it can not determine the verbal primitive corresponding to *runs on*, due to the fact that the concept *minicomputer* is unknown. Using, as a guide, the formation rules associated with verb *runs*, SNOWY asks for clues that may lead to the disambiguation of the verb. In this case, it determines that the primitive corresponding to *runs on* is *drive*. A similar situation occurs when SNOWY examines the noun phrase "3 megabytes of memory". In this case, SNOWY obtains from the user the fact that *megabyte* is a *unit of measure*.

>> > scheme runs on pcs and takes 1 megabyte of memory.
 :::unknown concept found: *pc*
 :::assuming it is a kind of *machine*
 :::is this a valid assumption?

> > > yes, pcs are microcomputers.
 :::in the following hierarchy, which of the categories is the most appropriate for
 microcomputer?

```
machine
  computer
    minicomputer
  engine
```

> > > computer.
 :::integrating *microcomputer* as a subconcept of *computer*

REMARKS: The verb phrase *runs on* is found for a second time in this sentence and, as in the first case, SNOWY is unable to determine the corresponding primitive due to the fact that the concept *pc* is unknown. In this case, though, SNOWY assumes that *runs on* has the same meaning as before, and therefore, that "pcs are machines", and requests the user to confirm this. The user answers affirmatively and goes on to specify that "pcs are microcomputers". Since *microcomputer* is an unknown concept, SNOWY displays the current hierarchy under the concept *machine* and asks the user to select the most adequate category for the new concept. In any case, the user may override SNOWY's assumptions by providing a negative answer or by indicating that none of the choices given are appropriate. In future implementations, much of this interaction will be done through a graphic interface with the help of a mouse.

> > > common-lisp and golden-hill are versions of lisp.
 > > > golden-hill runs on microcomputers.
 > > > golden-hill requires 2 megabytes of memory.
 > > > quintus and turbo are versions of prolog.
 > > > turbo runs on pcs and requires 2 megabytes of memory.
 > > > (showc language)

```
language
  programming-language
    dmy0 ((is-a(programming-language)) (manipulate(symbol)))
  lisp
    common-lisp
    golden-hill
    franz
    scheme
  prolog
    quintus
    turbo
  numeric-language
    fortran
```

REMARKS: The command "showc" displays the hierarchy of concepts that has been built under the concept *language*. Programming languages are categorized as *programming languages which manipulate symbols*, called *symbolic-languages*, and *numeric-languages*. Once SNOWY has acquired some knowledge from a domain expert, a user of the system may interact with it, in this case, to select a programming language. The following questions illustrate this.

```
>>> what kinds of programming languages are there?
::symbolic-language is-a programming-language
::numeric-language is-a programming-language

>>> what symbolic languages are there?
::lisp is-a <programming-language manipulate symbol>
::prolog is-a <programming-language manipulate symbol>
::common-lisp is-a lisp
::golden-hill is-a lisp
::franz is-a lisp
::scheme is-a lisp
::quintus is-a prolog
::turbo is-a prolog
```

REMARKS: In response to this question, SNOWY displays all languages that are symbolic languages. The concept *symbolic-language* is stored in LTM under the dummy name *dmy0*, and is defined by its *characteristic-features* "is-a programming-language" and "manipulate symbols." Thus, a question like "what languages manipulate symbols?" would have the same answer.

```
>>> which of these languages run on pcs?
::scheme drive pc
::turbo drive pc

>>> how much memory is required by scheme?
::scheme occupy 1 (megabyte) memory

>>> is scheme used for ai applications?
::yes

>>> is turbo used for ai applications?
::yes

>>> what is the difference between scheme and turbo?
::scheme used-for general-ai-application
::turbo property backtracking
```

Appendix II

PARSING AND FORMATION OF CONCEPTS

In this appendix we show the parser and formation phase outputs for sample sentences. Symbols of the form *gxx* denote structures built by the parser,

corresponding to sub-clauses within the main clause, @xxx denote *action-structures*, while the symbols dmyxx denote *object-structures* having a *characteristic-features* slot.

Given the sentence

programming languages which manipulate symbols are used for ai applications
the parser returns the structures:

```
g76
(subj(somebody) verb (are used) obj ((adj programming) (noun languages))
  rela g77 prep (for ((noun ai) (noun applications)))
g77
(subj ((adj programming) (noun languages)) verb (manipulate) obj ((noun
  symbols)))
```

These structures are passed on to the formation phase, which builds the representation structures below.

```
((dmy0(cf(is-a(programming-language)) (manipulate (symbol (q2(?))))))
  (manipulate (symbol ($more (@a1))))
  (used-for (ai-application (q1 (?)) (q2 (?))))
  (@a1 (args (dmy0) (symbol)) (pr (manipulate)) (q1 (all)) (q2 (?)))
  (symbol (manipulate%by (dmy0 ($more (@ai))))
  (ai-application (used-for%by (dmy0)) (is-a (application))))
```

The concept *programming languages which manipulate symbols* is represented by the *object-structure* dmy0, which contains a *characteristic-features* slot. Some of the quantifier slots contain a question mark to indicate that the quantifier is unknown. These structures become the input to the Integration Phase, which links the new concepts to the concepts currently in LTM by defining the *is-a* and *classes-of* slots.

As a second example, consider the sentence

scheme runs on pcs and takes 1 megabyte of memory

The parser output is

```
g434
(subj ((noun scheme)) verb (runs-on) obj ((nouns pcs)) conj (and) link g344)
g344
(subj (subj-of-prev) verb (takes) obj ((num 1) (noun megabyte)) prep (of((noun
  memory))))
```

which the Formation Phase transforms to

```
((scheme (drive (pc ($more (@a4))))
  (occupy (memory (q2 (1 (megabyte))) (q1 (?))))
  (pc (drive%by (scheme ($more (@a4))))
  (@a4 (args (scheme) (pc)) (pr (drive)) (q1 (?)) (q2 (?)))
  (memory (occupy-by (scheme (q1 (1 (megabyte))) (q2 (?))))))
```

Appendix E

Classification-Based Reasoning

Fernando Gomez
Department of Computer Science
University of Central Florida
Orlando, FL 32816
gomez@ucf.csnet

Carlos Segami
Department of Mathematics and Computer Science
Barry University
Miami Shores, FL 33161

Abstract

A representation formalism for N-ary relations, quantification and definition of concepts is described. Three types of conditions are associated with concepts: (1) necessary and sufficient properties, (2) contingent properties and (3) necessary properties. It is also explained how complex chains of inferences can be accomplished by representing existentially quantified sentences, and concepts denoted by restrictive relative clauses as classification hierarchies. The representation structures which make possible the inferences are explained first, followed by the reasoning algorithms which draw the inferences from the knowledge structures. All the ideas explained in this paper have been implemented and are part of the information retrieval component of SNOWY, a program which understands scientific paragraphs. The paper ends with an appendix containing a brief session with the program.

This is a modified and highly extended version of the paper "Classification-Based Inferences in Retrieving Information from a Database of Scientific Facts" which appeared in *Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications*, Miami, Florida, 1989.

This work was supported by NASA, Kennedy Space Center, under grant No. NAG-10-0058.

1. Introduction

For the last few years, we have been working on a model of comprehension of scientific texts [7]. We have also built a computer program, called SNOWY, which embodies this model [9]. SNOWY is a knowledge-lenient model of comprehension which reads texts with minimal knowledge or no knowledge at all about their contents. This aspect differentiates SNOWY, which builds conceptual representations from scratch by using formation, or learning, rules, from knowledge-intensive models of comprehension based on the notions of plans [21], MOPs [18] or TAU's [5]. Also, our work differs from recent research on discourse which has focused on dialogue or conversation [13,1,16,12,20].

The system has the following components: parsing, forming, recognizing, finding explanations, and integrating concepts into long-term memory (LTM). The Formation Phase builds from the output of the parser [8] the LTM knowledge representation structures used by the system. The Recognition Phase determines which concepts and relations built by the Formation Phase are known to the system. The Explanation Phase tries to understand the explanatory links connecting sentences (e.g. Antibiotics work against infections. They kill the germs which cause them.), and the Integration Phase integrates into LTM the concepts and relations which the Recognition Phase fails to recognize.

In [10], we explained briefly how this paradigm of comprehension differs from knowledge-intensive models of comprehension and, then, we went on to explaining in depth the recognition, classification and integration algorithms. In this paper, we describe the knowledge representation structures and the inference algorithms which allow SNOWY (1) to answer questions posed by users. In particular, in this paper we pay special attention to the representation of quantification, N-ary relations and the definition of concepts. Likewise, special emphasis has been put in showing that the algorithms described in this paper derive valid inferences.

The key concept on which SNOWY relies to draw inferences is the notion of *classification*. Consider the following paragraph below:

All whales are animals. All animals which live in the Antarctic eat fish. Some whales live in the Antarctic.

These sentences can be represented in Predicate Calculus as (the existential quantifier is represented as $\exists x$, and the universal quantifier as $\forall x$):

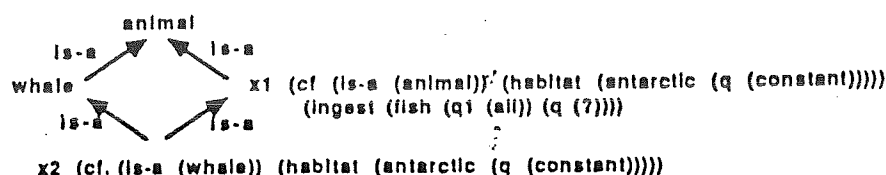
1. $\forall x (Wale(x) \rightarrow Animal(x))$
2. $\forall x (Animal(x) \rightarrow (\exists y (Live(x, Antarctic) \rightarrow \forall y (Fish(y) \rightarrow Eat(x, y))))$
3. $\exists z (Whale(z) \wedge Live(z, Antarctic))$

Suppose that given the above paragraph, the following question is asked: Do whales eat fish?"

The Predicate Calculus representation of the question is $\forall x \forall y (Whale(x) \wedge Fish(y) \wedge Eat(x, y))$.

Using resolution to answer the question is beyond the point, because the database of facts does not consist of just the facts listed above, and resolution does not offer any indexing mechanism to access the right chunk of knowledge.

Yet, the inference can be drawn relatively easily if restrictive relative clauses and existentially quantified sentences are represented as classification hierarchies, and a classification algorithm is invoked to integrate the newly created concepts in the right place in long-term memory (LTM). The representation of restrictive relative clauses and existentially quantified sentences as classification hierarchies is one of the major aspects which distinguishes our approach from KL-ONE [2,19]. Other differences are the representation of N-ary relations, quantification and the association of three types of properties with a given concept: (1) clusters of necessary and sufficient conditions, (2) necessary conditions and (3) contingent conditions. The inference algorithms which we will discuss in the next sections are a direct consequence of this representation. In our example, the clause "all animals which live in the Antarctic" will be represented as a subclass of *animals* distinguished by the fact that they live in the Antarctic; the sentence "whales are animals" will be integrated as "whale -- is-a --> animal;" and, when the sentence "some whales live in the Antarctic" is read, a subconcept of *whale* is created, namely, the concept denoted by "those whales which live in the Antarctic." Then, when the concept *whales which live in the Antarctic* is integrated in LTM, the classification algorithm determines that it must be integrated as a subconcept of the concepts *whale* and *animals which live in the Antarctic*, as illustrated in the diagram below. (The meaning of the terms used in the diagram is explained in Sections 2 and 3.) With this hierarchy of



concepts in LTM, the answer to the question "do whales eat fish" can be obtained by examining the concept *whale*, then descending to the concept *whales which live in the Antarctic* and ascending to the concept *animals which live in the Antarctic*. (The precise way in which this done is explained in Section 5, Table 2.)

In the example we have just seen the answer is obtained by examining the concepts already in LTM. However, there is a type of inference which takes place when a question asks information about a concept which does not exist in LTM. For instance, suppose that the facts referred by the following sentences are represented in LTM:

All birds are animals. All animals which live in the Antarctic swim.

The question is "do birds which live in the Antarctic swim?" In this case, the answer to the question can not be found in LTM, because the concept *birds which live in the Antarctic* does not exist in LTM. The only way in which we or the system could have formed the concept *birds which live in the Antarctic* is if we had been told that "birds live in the Antarctic." However, the answer can be obtained by activating the *Classifier* algorithm and asking where the concept *birds which live in the Antarctic* would be placed in LTM, if it existed there. The *Classifier* will link the concept *bird which live in the Antarctic* to the upper-concept *animal which live in the Antarctic*, and the answer to "do birds which live in the Antarctic swim?" can be obtained by Inheritance.

In order to set up the stage to properly discuss the reasoning algorithms, we need to explain the knowledge representation structures used by SNOWY. This is a necessary component of every paper dealing with SNOWY's algorithms since these depend on the knowledge representation structures. However, in this paper we have provided First Order Predicate Calculus (FOPC) formulae to make clear the meaning of the representation structures. Sections 2 and 3 deal with the knowledge representation structures, the representation of restrictive clauses and existentially

quantified sentences. In section 4, we explain is-a inheritance in SNOWY. In sections 5 and 6, we explain the reasoning techniques based on our representation structures. In Section 7, we explain briefly the types of questions handled by SNOWY. In section 8, we give our conclusions. Finally, we have included an appendix containing a brief session with SNOWY.

2. Representation of Concepts In LTM

This section explains the representation of concepts in LTM, so that the reader can follow the subsequent sections. In order for SNOWY to understand a text, it needs to start with at least a minimum set of concepts, which categorizes the world into states, actions, collections, etc. These concepts are organized in a hierarchy (Figure 1). Generic concepts in the hierarchy are linked through *is-a* and *classes-of* slots. All of these concepts denote non-empty sets of individuals. Indi-

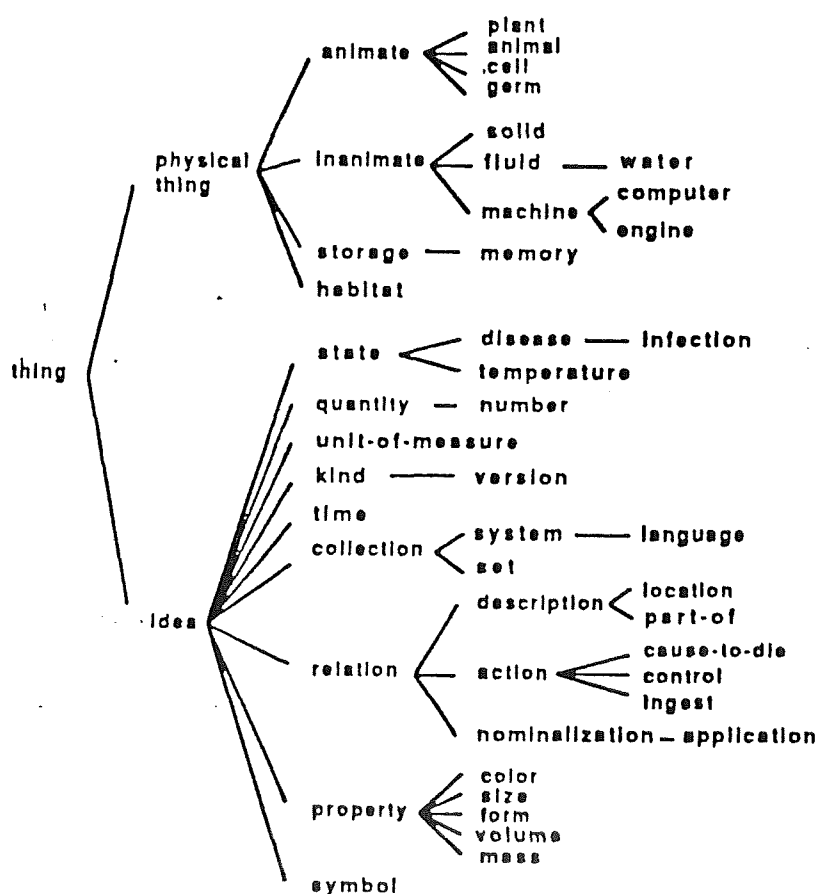


Figure 1. Partial Contents of the Initial Hierarchy of Concepts in LTM

vidual members of a class, such as Peter or Fido, are linked to their parent concepts through *instance-of* and *instances* slots. As SNOWY starts reading a text and begins adding new concepts to the hierarchy, this becomes a "tangled" hierarchy. New concepts are recognized and integrated in this *a priori* hierarchy. The concepts in the hierarchy are represented using three kinds of representation structures: *object-structures*, *action-structures*, and *event-structures*. Table 1, below, describes the syntax of *object-structures* and *action-structures*. The syntax of *event-structures* is the same as that of *action-structures*, except that the slot "(instance-of (event))" is always present in *event-structures*.

An *object-structure* contains knowledge about physical or abstract objects. As shown in Table 1, an *object-structure* consists of the object's name, an optional *cf* slot (explained in detail in Section 3), and one or more slots called conceptual relations or attributes. Two kinds of conceptual relations can be distinguished: those which describe objects and those that attribute actions to physical things. This distinction corresponds to the distinction between descriptive and action verbs in natural language [6]. "Hormones are chemicals" and "all centipedes are slim and have many legs" are descriptive sentences, while "antibiotics kill germs" and "cells take in sugar" express actions performed by physical things. The knowledge in the sentence about centipedes is represented as:

```
(centipede
  (form (slim (q1 (all))))
  (has-body-part (leg (q1 (all)) (q (many)))))
```

This example corresponds to grammar rules 1, 2, 4, and 5 for *object-structures*, in Table 1. The nesting of the quantifiers in the structure is from left to right. Thus, the meaning of the second relation in the *object-structure* for *centipede* is (Since every concept in LTM denotes a non-empty set of individuals, the FOPC formula below should be preceded/followed by " $\forall x(\text{Centipede}(x))$ " and " $\forall y(\text{Leg}(y))$." We postpone the discussion of this until the next section.):

$$(x) (\text{Centipede}(x) \rightarrow \forall y(\text{Leg}(y) \text{ and } \text{Has-body-part}(x,y)))$$

The slot *q1* contains the quantifier of the concept described by the *object-structure*, and the slots *q* contain the quantifier of the other arguments of the relation. Note that because FOPC does not have a quantifier for "many" we have represented "many" as an existential quantifier. (These

Object Structures

```

<object-struct> ::= ( <object-name> [ <cf-slot> ] { <slot> }+ )
<slot>          ::= <action-slot> | <desc-slot>
<action-slot>   ::= ( <action-relation> { ( [ <argument> ] { <action-struct-name> }+ ) }+ )
<desc-slot>     ::= ( <descriptive-relation> { <desc-arg> }+ )
<desc-arg>      ::= ( <object-name> [(q1 (<quantifier>))] [(q (<quantifier>))]
                        { ( <relation> { ( <object-name> (q (<quantifier>))) }+ ) } )
<cf-slot>       ::= ( cf (is-a ((object-name)))+ ) { <desc-slot> } { <cf-action> } )
<cf-action>     ::= ( <action-relation> (( [ <object-name> (q (<quantifier>))] { <case-slot> } ) ) )

```

Action Structures

```

<action-struct> ::= ( <action-struct-name> <args-slot> <relation-slot> { <case-slot> }+ )
<args-slot>      ::= ( args { ( <argument> ) }+ )
<relation-slot>  ::= ( pr ( <action-relation> ) )
<case-slot>      ::= ( <case-name> ( <object-name> ( q ( <quantifier> ) ) ) ) |
                      ( <case-name> ( <event-struct-name> ( q (some))) )
<argument>       ::= <object-name> | <event-struct-name>
<action-struct-name> ::= a1 | a2 | .....
<event-struct-name>  ::= e1 | e2 | .....
<object-name>        ::= animate | mammal | .....
<action-relation>    ::= ingest | make | .....
<quantifier>         ::= all | some | constant | many | most | few | .....
<case-name>          ::= source | destination | instrument | .....

```

Table 1. The Syntax of Object-Structures and Action-Structures

NOTES: (1) The notation {A} denotes zero or more occurrences of the string A, {A}⁺ denotes one or more occurrences of A, while [A] denotes zero or one occurrence of A. (2) The syntax for *event-structures* is the same as *action-structures*, except that the slot "(instance-of (event))" must always be present in *event-structures*.

Issues are discussed in detail in the section below.) Conceptual relations denoting actions are represented in the *object-structure* as:

(concept1 (action-relation (a1))),

If the relation takes no object, while relations that do take an object are represented as:

(concept1 (action-relation (concept2 a2)))

where *concept2* is the object of the relation. If the same relation is true of two or more concepts, it is represented as:

(concept1 (action-relation (concept2 a2) (concept3 a3) ... (concept*i* a*i*)))

Thus, the relation underlying the sentence "Germs attack animals and plants" is represented in the *object-structure* of *germ* as:

(germ (attack (animal a2) (plant a3)))

This example corresponds to grammar rules 1, 2, and 3 for *object-structures*, in Table 1. The terms *a1*, *a2*, ... *a*i** stand for the names of the conceptual relations. The structure which represents the conceptual relation itself is called an *action-structure*. For example, the *action-structure* representing the conceptual relation in the sentence "bacteria cause diseases" is as follows:

(a1	(Inst a1 cause)
(args (bacteria) (disease))	(actor a1 bacteria)
(pr (cause))	(object a1 disease)
(actor (bacteria (q (?))))	(quantifier-subject a1 unknown)
(object (disease (q (?))))	(quantifier-object a1 unknown)

Of these two representations, we will use the one on the left side. The one on the right side is a slot-assertion notation [4]. The slot *args* contains the arguments of the relation. If the relation is monadic, *args* will contain one concept. If the relation is diadic (as in this case), *args* contains two concepts, and so on. The slot *pr* contains the relation. The slots *q* contain the quantifier of the argument where the slot is embedded. A question mark in a quantifier slot means that the contents of *q* may be a universal quantifier or an existential quantifier. When an argument of a relation is an individual member of a class (i.e., Peter, Fido) its quantifier slot will contain the value *constant*, to indicate that it is a member of a class (see for example *Antarctic* in the diagram of page 3). Note that necessary conditions are associated with a concept since the value of a

(bacteria (cause (disease a1)))	(a1
	(args (bacteria)(disease))
	(pr (cause))
(disease (cause:by (bacteria a1)))	(actor (bacteria (q (?))))
	(object (disease (q (?))))

Figure 2. The Representation of "bacteria cause diseases"

(blood (transport (oxygen a1)))	(a1
	(args (blood)(oxygen)(cell))
	(pr (transport))
(oxygen (transport:by (blood a1)))	(actor (blood (q (?))))
	(object (oxygen (q (?))))
(cell (destination-of (a1)))	(destination (cell (q (?))))

Figure 3. The Representation of "Blood transports oxygen to cells"

quantifier can be "all" in an *action-structure*.

The representation of "bacteria cause diseases" is done by using the *object-structures* and *action-structure* which are displayed in Figure 2. Every concept which fills a case (actor, object, destination, source, etc.) in the sentence is indexed independently, with pointers to the *action-structure* (conceptual relation) of which they are part. In the above example, both the actor, *bacteria*, and the object, *diseases*, are indexed using an *object-structure*. Note how the concepts *diseases* and *bacteria* point to the same *action-structure*. Similarly, Figure 3 shows the representation of the relation underlying the sentence "Blood transports oxygen to cells." This is a relation with three arguments, *blood*, *oxygen*, and *cells*. Each of them has its own *object-structure* in LTM, and each one of these structures points to the *action-structure* representing the given relation.

The Meaning of Action-Structures

Consider the sentence "all people like dogs." The *action-structure* for the representation of this sentence is:

```
(a1
  (args (human)(dog))
  (pr (like))
  (actor (human (q (all))))
  (object (dog (q (some))))
```

The meaning of *action-structures* similar to the above, for different values of the quantifiers, is given below. (We denote the universal quantifier with (x) and the existential quantifier with Vx.) Also, we assume that the implication connective has the lowest precedence followed by "or" and "and."

1. (a1

(args (human)(dog))
(pr (like))
(actor (human (q (all))))
(object (dog (q (all))))

(x)(y) (Human(x) and Dog(y) \rightarrow Like(x,y)) and Vx(Human(x)) and Vy(Dog(y))

2. (a1

(args (human)(dog))
(pr (like))
(actor (human (q (all))))
(object (dog (q (some))))

(x) (Human(x) \rightarrow Vy (Dog(y) and Like(x,y))) and Vx(Human(x)) and Vy(Dog(y))

3. (a1

(args (human)(dog))
(pr (like))
(actor (human (q (some))))
(object (dog (q (all))))

Vx(Human(x) and (y) (Dog(y) \rightarrow Like(x,y))) and Vy(Dog(y))

4. (a1

(args (human)(dog))
(pr (like))
(actor (human (q (some))))
(object (dog (q (some))))

VxVy(Human(x) and Dog(y) and Like(x,y))

The FOPC formulas for structures (1), (2) and (3) are followed by Vx(Human(x)) and Vy(Dog(y)) because in our representation every concept in LTM denotes a non-empty set of individuals. The formula (x)(y) (Human(x) and Dog(y) \rightarrow Like(x,y)) is not a correct representation of structure (1), since this formula is true even if there are no humans or dogs. Therefore, our representation of "all Greeks liked unicorns" includes *unicorns* as entities in our ontology, since these putative entities will be under the scope of an existential quantifier [18]. Then, how will the algorithm know that unicorns did not live? This could be done by creating a category in our *a priori* hierarchy of entities which do not have physical existence, and properly indexed under them will be such things as

unicorns, fairies, etc. So, if SNOWY were asked the question: Were there unicorns? the answer will be "Of course, and all Greeks liked them," but they do not have a physical existence.

From here on, we will omit the existentially quantified predicates (at the end of the formulas) which indicate that concepts represented by *object-structures* denote non-empty sets of individuals.

As one can see, the nesting of the quantifiers is from left to right. Hence, the *action-structures* above can not represent the following two statements: "There is a dog which is loved by everybody" and "Every dog is loved by somebody," which correspond to the two FOPC formulas below, respectively.

5. $\forall y(\text{Dog}(y) \text{ and } (x) (\text{Human}(x) \rightarrow \text{Like}(x,y)))$

6. $(y) (\text{Dog}(y) \rightarrow \forall x(\text{Human}(x) \text{ and } \text{Like}(x,y)))$

In fact, formulas (3), (4) and (5) will be not be represented using an *action-structure*. Since they correspond to existentially quantified sentences, they are are represented as classification hierarchies, as indicated in the introduction and explained in detail in the next section. Formula (6) above is represented by using the inverse relation of *like*, namely *like.by*, in the following way:

(human	(a1
(like (dog a1)))	(args (dog) (human))
	(pr (like:by))
(dog	(actor (dog (q (all))))
(like:by (human a1)))	(object (human (q (some))))

The meaning of *a1* is:

$(y)(\text{Dog}(y) \rightarrow \forall x(\text{Human}(x) \text{ and } \text{Like:by}(y,x)))$

Consider the following structure in LTM:

```
(a1
  (args (whale)(fish))
  (pr (Ingest))
  (actor (whale (q (?))))
  (object (fish (q (all))))
```

The question mark in the quantifier slot of concept *whale* means that the content of *q* may be a universal quantifier or an existential quantifier. Then, the structure above may mean:

$(x)(y) (\text{Whale}(x) \text{ and } \text{Fish}(y) \rightarrow \text{Eat}(x,y))$

or,

$$\forall x(\text{Whale}(x) \text{ and } (y) (\text{Fish}(y) \rightarrow \text{Eat}(x,y)))$$

If in a chain of inferences the reasoning algorithm comes across an argument (in a relation) which is quantified with a question mark, it interprets the question mark as an existential quantifier. This is consistent with not amplifying the premises or conclusion in an inference. Other quantifiers such as "many," "most," "few," etc. are also taken as existential quantifiers in drawing deductive inferences. Yet, these quantifiers play an important role in the inductive learning algorithms which we are presently constructing for SNOWY.

In summary, the semantics of our representation structures is obtained by translating them into FOPC formulae (the nesting of quantifiers is from left to right). Any interpretation that makes true the FOPC formula makes also true our corresponding knowledge representation structure.

The Representation Of Embedded Relations

A relation which is an argument of another relation is represented using an *event-structure*. In the sentence, "insulin causes cells to take in sugar" the conceptual relation underlying the clause "cells take in sugar" is represented using an *event-structure*. From a representational point of view, *event-structures* are like *action-structures*, except that they contain the slot "instance-of (event)." The important thing to note here is that relations represented by *event-structures* can not be concluded to be true separate from the *action-structures* in which they are embedded. For instance, from "alcohol causes most people to do foolish things," it can not be concluded that "most people do foolish things." The representation of "insulin causes most cells to take in sugar" (see Figure 4) is done by using an *object-structure*, an *action-structure* and an *event-structure* (assuming that "insulin" is universally quantified). In all of our examples so far, the object of an action has been a concept denoting entities. However, in the sentence "insulin causes most cells to take in sugar" the object is an event. Thus, in Figure 4, the *object-structure* is (Insulin (cause(e1 a1))), where the object of the relation *cause* is the relation represented by the *event-structure* e1. Therefore, the *action-structure* a1, which represents the relation *cause*, has e1 in its *args* and *object* slots. The question "what does insulin cause" is answered by examining the *action-structure* a1 and retrieving the object, which in this case is the structure e1: "most cells ingest

sugar." The meaning of the representation in Figure 4 is: (Note that the meaning of "most" is lost in the FOPC formula.)

$(x)(\text{Insulin}(x) \rightarrow \forall y \forall z (\text{Cell}(y) \text{ and } \text{Sugar}(z) \text{ and } \text{Cause}(x, \text{Ingest}(y, z))))$

Prior to discussing how quantifiers in the embedding relation and the embedded relation relate one to another, consider the sentences below:

- (1) Everyone wants to own a house.
- (2) Everyone wants everyone to own a house.

From (1), it can be concluded that Peter wants to own a house, but not that Peter wants Mary to own a house. However, both inferences can be concluded from (2). The representation of (1) and (2) are depicted in Figures 5 and 6, respectively. In the representation of (1), the universal quantifier "all" has an index, x , in the *action-structure* and in the *event-structure*. This is so, to indicate that the concept "human" in the *event-structure* is the same as the one in the *action-structure*. The FOPC formulas for the structures in Figures 4 and 5 are, respectively:

$(3) (x)(\text{Human}(x) \rightarrow \forall y (\text{House}(y) \text{ and } \text{Want}(x, \text{Own}(x, y))))$

(Insulin (cause (e1 a1)))

(e1	(a1
(Instance-of (event))	(args (Insulin)(e1))
(args (cell)(sugar))	(pr (cause))
(pr (Ingest))	(actor (Insulin (q (all))))
(actor (cell (q (most))))	(object (e1 (q (some))))
(object (sugar (q (?))))	(Instrument (?))

Figure 4. The Representation of "insulin causes most cells to take in sugar"

(a1	(e1
(args (human)(e1))	(Instance-of (event))
(pr (want))	(args (human)(house))
(actor (human (q (allx))))	(pr (own))
(object (e1 (q (some))))	(actor (human (q (allx))))
	(object (house (q (some))))

Figure 5. The Representation of "everyone wants to own a house"

<pre>(a1 (args (human)(e1)) (pr (want)) (actor (human (q (all)))) (object (e1 (q (some))))))</pre>	<pre>(e1 (instance-of (event)) (args (human)(house)) (pr (own)) (actor (human (q (all)))) (object (house (q (some))))))</pre>
--	---

Figure 6. The Representation of "everyone wants everyone to own a house"

(4) $(x)(\text{Human}(x) \rightarrow \forall y(\text{House}(y) \text{ and } (z)(\text{Human}(z) \rightarrow \text{Want}(x, \text{Own}(z,y))))))$

An indexed quantifier may appear in an *action-structure* without any *event-structures* embedded in it, as in the sentence "everyone likes himself/herself" whose representation is:

```
(a1
  (args (human)(human))
  (pr (like))
  (actor (human (q (allx))))
  (object (human (q (allx))))))
```

3. Representing Concepts Denoted by Restrictive Relative Clauses and Existentially Quantified Sentences as Classification Hierarchies

The representation of concepts denoted by restrictive relative clauses and other restrictive qualifiers such as prepositional phrases involves their appropriate integration in the hierarchy. Concepts such as *hormones produced by the body*, *germs which enter the body* or "cars with fuel injection" are described by complex descriptions. Yet, these concepts need to have a unique name in memory so that knowledge about them can be integrated under the same node. These concepts are represented in LTM via an *object-structure* with a dummy name (a gensym), and their representation is characterized by the presence of a *characteristic-features* slot whose purpose is to describe the features that are characteristic of the concept. For example, when building the representation of the sentence, "insulin is a hormone produced by the pancreas," the concept, *hormones produced by the pancreas*, a subconcept of the concept *hormones*, is created with the following representation:

```
(x1 (cf (is-a (hormone)) (make:by (pancreas (q (?))))))
(hormones (classes-of (x1)))
```

This concept is identified by the contents of the *characteristic-features* slot and not by the dummy

name $x1$. Thus, if a question like "what do hormones produced by the pancreas cause?" is asked, the system recognizes that the concept "hormones produced by the pancreas" was previously created and accesses the *object-structure* $x1$. (The slot q contains "?" because it is unknown if every pancreas makes hormones.)

The meaning of the *cf* slot above can be expressed by the following FOPC (we are assuming that the value of the question mark is "some").

$(x) (X1(x) \text{ iff } (\text{Hormone}(x) \text{ and } \forall y(\text{Pancreas}(y) \text{ and } \text{Make:by}(x,y)))$

From a formal point of view, the *cf* slot can be viewed as playing the same role as a meaning postulate [3]. The *cf* slot identifies a concept by providing necessary and sufficient conditions. At least one of these conditions must a parent concept.

Complex noun groups such as *sea mammals* and nouns modified by prepositional phrases such as *birds with long legs* are represented in a similar form. The concept *sea mammal* is represented as:

$(x1 \text{ (cf (Is-a (mammal)) (habitat (sea (q (?))))))$
 $(\text{mammal (classes-of (x1)))}$

How the category "habitat (sea)" is computed does not concern us in this paper.

The Representation of Existentially Quantified Sentences

Sentences which begin with an existential quantifier such as "some mammals live in the sea" require a representation similar to the one we have proposed for restrictive relative clauses. Consider the sentence "some mammals live in the sea." If an unique node in memory is not created for the concept *mammals which live in the sea* then subsequent knowledge about this concept could not be integrated. Creating two object structures "mammal" and "sea" linked to an *action-structure* containing the relation *habitat*, quantifier of the actor = some, and quantifier of the object = ?, will not do any good if the next sentence is "these animals are in great danger." The conceptual relation underlying this concept could not be stored under the node corresponding to *mammals which live in the sea*, since this concept would not exist in LTM. Hence, the representation of the sentence "some mammals live in the sea" is:

```
(x1 (cf (is-a(mammal)) (habitat(sea (q (?))))))
(mammal (classes-of (x1)))
```

Note that because every concept in LTM denotes a non-empty set of individuals, the existence of mammals is captured in this representation. This representation of existentially quantified sentences as classification hierarchies is consistent with the use of such sentences in expository discourse. Constructions like "Some programming languages are used for numeric computation. Numeric programming languages" and many others similar to this abound in expository prose. It will take us well beyond the limits of this paper to provide further details to back up this view (see [11]).

4. Is-a Inheritance

Prior to our discussion of inheritance in SNOWY, we introduce the following definitions.

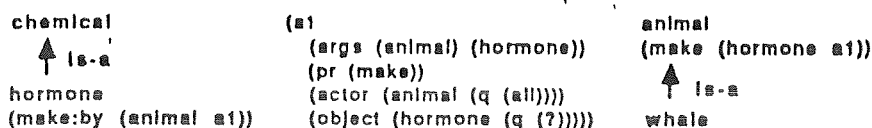
Definition 4.1

Let x and y be concepts in LTM. Concept y is said to be a "proper subclass" of x if there exists a sequence of concepts y_1, y_2, \dots, y_n in LTM, such that $y_1 = y$, $y_n = x$, and the relation " y_i is-a y_{i+1} " exists in LTM for all $i = 1, \dots, n-1$. Concept y is said to be a "subclass" of x if either $y = x$, or y is a proper subclass of x . If y is a proper subclass of x , then y is also called a "descendant" or "sub-concept" of x , and x is called an "ancestor" or "superconcept" of y .

There are two ways in which inheritance mechanisms work in our representation language: (1) by means of *is-a* relations combined with universal quantification and (2) through *characteristic-features* slots. Let us suppose that the following two sentences are read:

All animals produce chemicals called hormones. Whales are animals.

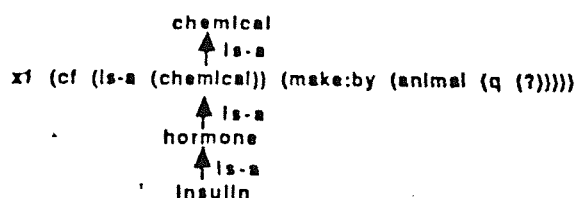
The concepts and relations underlying these sentences are integrated in LTM as follows:



If the question "do whales produce hormones?" is asked, the concept *whale* is examined first,

looking for the conceptual relation "whale make hormone." Because the conceptual relation is not found there, the concepts in the *is-a* slot of *whale* are examined next. Since the conceptual relation "animal make hormone" is found under the concept *animal*, the *action-structure* *a1* will be searched for the contents of the quantifier *q1*. If this quantifier is found to be "all," as in this case, the answer to the question is "yes." However, if the first sentence had been "animals produce chemicals called hormones," the quantifier *q1* in the *action-structure* *a1* would not be "all," and the question "do whales produce hormones?" would have been answered "I don't know." Thus, the descendants of a concept inherit only those conceptual relations in which the concept is universally quantified.

The second case is inheritance through *characteristic-features* slots. To illustrate this case, we use an example for which an *object-structure* containing a *characteristic-features* slot is built. Suppose the sentences "All hormones are chemicals produced by animals," and "Insulin is a hormone" are read. The underlying concepts and relations are integrated as:



Note that the representation of "All hormones are chemicals produced by animals" is obtained by first building the concept *chemicals produced by animals*, denoted by *x1* in the figure above, and then integrating *hormone* as a subconcept of *x1*. The concept *insulin* is then integrated as a subconcept of *hormone*. Given this, suppose the question "is insulin made by animals?" is asked. The answer will be "yes," because the conceptual relation "*x1* make:by (animal)" is a characteristic feature of the class *x1*, and all subclasses of *x1* inherit such conceptual relations. However, if the question is "is insulin made by all animals," the answer is "I don't know."

We formalize the ideas in the previous examples with the following definition.

Definition 4.2

Let "*B1 r B2, B3,..., Bk*" denote a conceptual relation, where *B1,...,Bk* denote concepts, and *r* denotes a relation whose arguments are *B1,...,Bk*. Suppose that this relation exists in LTM.

(a) Let the concept A be a proper subclass or an instance of B1.

(1) If B1 is universally quantified in the relation "B1 r B2, B3,..., Bk," or if "B1 r B2, B3,..., Bk" is a characteristic feature of B1, then concept A inherits the relation "A r B2, B3,..., Bk" from B1.

(2) If B1 has an Indexed universal quantifier in the relation "B1 r B2, B3,..., Bk," or in the characteristic feature "B1 r B2, B3,..., Bk," then A inherits from B1 the relation obtained from "B1 r B2, B3,..., Bk" by replacing with A each argument that has the same indexed quantifier as B1.

(b) If a concept A is a proper subclass or an instance of Bi for some i, i=2,3,...,k, then A inherits the conceptual relation "B1 r B2,...,B(i-1), A, B(i+1),..., Bk" from Bi, if Bi is universally quantified in the conceptual relation "B1 r B2, B3,..., Bk."

Case (a.2) above, is illustrated with the following example. If "all americans want to own a house," and "all texans are americans," then the concept "texans" inherits the relation underlying "all texans want to own a house" from the concept "americans."

5. Inferring the Answer from the Classification Links In LTM

In this section we study the kinds of inferences that may be drawn based on the classification links among concepts in LTM. For example, suppose that the following facts are known:

All infections are diseases
All bacteria are germs
Bacteria cause infections

If the question "Do germs cause diseases?" is asked, the system should answer affirmatively by reasoning as follows: "bacteria are germs, bacteria cause infections, and infections are diseases, so it can be inferred that some germs cause some diseases." Similarly, from the facts:

All red cells are cells
Sugar is a chemical
Insulin causes all cells to take in sugar

we would like to infer that "insulin causes all red cells to take in some chemicals," and from the facts:

All arctic whales are whales

All arctic whales are arctic mammals
 All arctic mammals feed on all arctic mollusks
 All arctic clams are arctic mollusks
 All arctic clams are clams

we would like to infer the fact that "some whales feed on some clams." All these inferences are based on the classification links among concepts. We introduce some criteria under which inferences of this type can be made, and show that these criteria lead to valid inferences.

Theorem 1

Let A_1, A_2, \dots, A_k and B_1, B_2, \dots, B_k be concepts in LTM, and let r be a relation. Let us assume that none of the arguments in the conceptual relation " $B_1 r B_2, B_3, \dots, B_k$ " has an indexed quantifier. Then, the conceptual relation " $B_1 r B_2, B_3, \dots, B_k$ " implies the conceptual relation " $A_1 r A_2, A_3, \dots, A_k$ " (or, the second one can be inferred from the first one), if for each $i, i=1, \dots, k$, either

- (1) $A_i = B_i$; or,
- (2) B_i is a proper subclass of A_i ; or,
- (3) A_i is a proper subclass of B_i and A_i inherits from B_i the relation " $B_1 r B_2, B_{(i-1)}, A_i, B_{(i+1)}, \dots, B_k$;" or,
- (4) There exists a concept C_i in LTM which is a proper subclass of both A_i and B_i , such that C_i inherits from B_i the relation " $B_1 r B_2, B_{(i-1)}, C_i, B_{(i+1)}, \dots, B_k$."

Furthermore, the quantifiers of A_1, A_2, \dots, A_k , in the conceptual relation " $A_1 r A_2, A_3, \dots, A_k$ " can be determined as follows:

- a) concept A_j is universally quantified if case (3) is true for A_j and B_j ;
- b) concept A_j is existentially quantified if case (2) or case (4) is true for A_j and B_j ;
- c) if case (1) is true for A_j and B_j , then A_j has the same quantifier that B_j has in the conceptual relation " $B_1 r B_2, B_3, \dots, B_k$."

Remark We note here that theorems 1, 2 and 3 will also hold if the term "a proper subclass of" is replaced with the term "a proper subclass of or an instance of," and the quantifier *constant* is treated as an existential quantifier. Thus, these theorems can also be applied when some of the arguments of the relations are constants. We do not explicitly include this case in the theorems

for the sake of simplicity.

Proof

We first prove the following lemma.

Lemma 1 Under the hypotheses of Theorem 1, for any $j, j = 1, \dots, k$, the relation

$$(I) \quad "B_1 r B_2, \dots, B_{(j-1)}, B_j, B_{(j+1)}, \dots, B_k"$$

Implies the relation

$$(II) \quad "B_1 r B_2, \dots, B_{(j-1)}, A_j, B_{(j+1)}, \dots, B_k"$$

where the quantifier of A_j in (II) is determined according to the criteria of Theorem 1.

Proof Four cases are possible.

(1) $A_j = B_j$.

In this case, clearly (I) implies (II), since they are identical, and the quantifiers of A_j and B_j are the same.

(2) B_j is a proper subclass of A_j .

To show that (I) implies (II), let us assume that relation (I) holds true, that is, that it exists in LTM.

If (I) holds true and B_j is a proper subclass of A_j , then (II) is true for some entities in the class A_j , namely those in B_j . Thus, (I) implies (II) with A_j existentially quantified.

(3) A_j is a proper subclass of B_j , and A_j inherits the relation (II) from B_j .

If (I) holds true, because of inheritance (I) must hold true for all entities in the class B_j and, in particular, for all entities in class A_j , since A_j is a proper subclass of B_j . Thus, (I) implies (II), and A_j is universally quantified in (II).

(4) C_j is a concept which is a proper subclass of both A_j and B_j , and C_j inherits the relation

$$(III) \quad "B_1 r B_2, \dots, B_{(j-1)}, C_j, B_{(j+1)}, \dots, B_k"$$

from B_j .

Applying case (3), above, to the conceptual relations (I) and (III), we conclude that (I) implies (III) with C_j universally quantified in (III). Applying case (2) to the conceptual relations (III) and (II), we conclude that (III) implies (II), with A_j existentially quantified in (II). Thus, (I) implies (II) with A_j

existentially quantified in (II). This completes the proof of Lemma 1.

Proof of the theorem

Suppose the conceptual relations " $B_1 \text{ r } B_2, B_3, \dots, B_k$ " and " $A_1 \text{ r } A_2, A_3, \dots, A_k$ " satisfy the hypotheses of the theorem. Then, applying Lemma 1 with $j = 1$, we conclude that the conceptual relation " $B_1 \text{ r } B_2, B_3, \dots, B_k$ " implies " $A_1 \text{ r } B_2, B_3, \dots, B_k$," where A_1 is quantified according to the criteria of the theorem. Next, we note that the hypotheses of Theorem 1 are satisfied by the relations " $A_1 \text{ r } B_2, B_3, \dots, B_k$ " and " $A_1 \text{ r } A_2, A_3, \dots, A_k$." Applying Lemma 1 to these relations, with $j = 2$, we conclude that " $A_1 \text{ r } B_2, B_3, \dots, B_k$ " implies " $A_1 \text{ r } A_2, B_3, \dots, B_k$." Therefore, " $B_1 \text{ r } B_2, B_3, \dots, B_k$ " also implies " $A_1 \text{ r } A_2, B_3, \dots, B_k$." Repeating this argument k times, we conclude that " $B_1 \text{ r } B_2, B_3, \dots, B_k$ " implies " $A_1 \text{ r } A_2, A_3, \dots, A_k$," where the quantifiers of A_1, A_2, \dots, A_k are determined by the criteria of the theorem. This concludes the proof of the theorem.

Answering Verification Questions

Let " $A_1 \text{ r } A_2, A_3, \dots, A_k$ " be the schema of a verification question, (e.g. "do germs enter the body?") where each A_i is a concept and r is a relation. The answer to the question is affirmative if there exists a concept B_1 in LTM whose *object-structure* contains the conceptual relation " $B_1 \text{ r } B_2, B_3, \dots, B_k$ " and " $B_1 \text{ r } B_2, B_3, \dots, B_k$ " implies the conceptual relation " $A_1 \text{ r } A_2, A_3, \dots, A_k$."

We now illustrate with examples some of the cases included in the criteria of theorem 1.

Example 1 Consider the following inference:

All infections are diseases
Some germs cause infections

Some germs cause some diseases

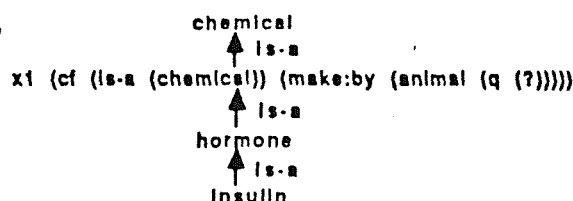
Here, the relations underlying the statements "all infections are diseases" and "some germs cause infections" are assumed to be represented in LTM. Thus, we have, $A_1 = B_1 = \text{germ}$, $A_2 = \text{disease}$, $B_2 = \text{infection}$, and B_2 is a subclass of A_2 . Because condition 2 of the theorem is true for A_2 and B_2 , A_2 is existentially quantified in the inferred relation.

Example 2

All hormones are chemicals produced by animals
 All insulins are hormones

All Insulins are produced by animals

The representation of the first two conceptual relations above, is illustrated in the figure below. Note how the concept "chemicals produced by animals" is built, and given the dummy name *x1*. In this example, *A1* = insulin, *A2* = animal, *B1* = *x1*, and *B2* = animal. Thus, *A1* is a subclass of *B1*, and *A1* inherits the relation "insulin make:by animal" from *B1* (*characteristic-features* slot inheritance).

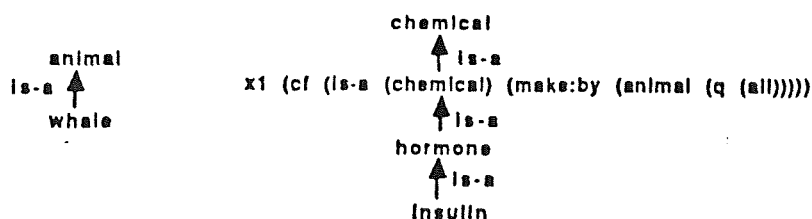


Example 3

All hormones are chemicals produced by all animals
 All whales are animals
 All insulins are hormones

All insulins are produced by all whales

The representation of the concepts and relations for the first three statements above, is illustrated as follows:



Here, *A1* = insulin, *A2* = whale, *B1* = *x1*, and *B2* = animal. Thus, *A1* is a subclass of *B1* and *A1* inherits the relation "insulin make:by animal" from *B1* due to the universal quantifier. Also, *A2* is a subclass of *B2* and it inherits the relation "*x1* make:by whale" from animal. Since condition 2 of the criteria of theorem 1 holds true for *A1* and *B1*, and *A2* and *B2*, *A1* and *A2* are both universally

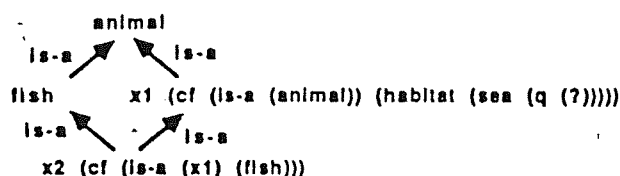
quantified in the inferred relation.

Example 4 This example illustrates inheritance in a "tangled hierarchy."

All sea fish are fish
 All sea fish are sea animals
 All sea animals live in the sea

Some fish live in the sea

Suppose that the system reads the sentence "some animals which live in the sea are fish." This sentence will be integrated as shown in the figure below. In order to answer affirmatively the question "do fish live in the sea?" we need to reach the concept $x1$ (sea animal) from "fish." Hence, we need to descend to the concept $x2$ (sea fish) and then ascend to $x1$. In this case, $A1 = \text{fish}$, $B1 = x1$, $A2 = B2 = \text{sea}$. If we take $C1 = x2$, then $C1$ is a subclass of both $A1$ and $B1$, and $C1$ inherits the relation " $C1$ habitat sea" from $B1$. Thus, $A1$ is existentially quantified in the inferred relation, and $A2$ has the same quantifier as $B2$.



Example 5

Blood transports oxygen from the lungs to all cells
 All oxygens are chemicals
 All muscle cells are cells

Blood transports some chemicals from the lungs to all muscle cells

In this case, $A1 = B1 = \text{blood}$, $A2 = \text{chemical}$, $B2 = \text{oxygen}$, $A3 = B3 = \text{lung}$, $A4 = \text{muscle cell}$, and $B4 = \text{cell}$. Thus, $B2$ is a subclass of $A2$, $A4$ is a subclass of $B4$ and $A4$ inherits the relation " $B1$ r $B2$ $B3$ $A4$ " from $B4$. In the inferred relation, $A2$ has an existential quantifier and $A4$ has a universal quantifier.

Theorem 1 deals with relations all of whose arguments are entities. In many instances, some arguments of a conceptual relation may be conceptual relations. For example, in the conceptual relation underlying the sentence "insulin causes all cells to take in sugar" the actor is a physical

object, while the object of the relation is a conceptual relation, i.e., the conceptual relation underlying the sentence "all cells take in sugar." From a statement like "insulin causes all cells to take in sugar" we would like to infer, for example, that "insulin causes all red cells to take in sugar." We now extend the criteria of Theorem 1 to handle the more general case when one or more arguments are conceptual relations.

Theorem 2

Consider the conceptual relations " $A_1 r A_2, \dots, A_k, A_{(k+1)}, \dots, A_n$ " and " $B_1 r B_2, \dots, B_k, B_{(k+1)}, \dots, B_n$," where A_1, \dots, A_k , and B_1, \dots, B_k denote entities, $A_{(k+1)}, \dots, A_n$ and $B_{(k+1)}, \dots, B_n$ denote conceptual relations, r is a relation, and the arguments in each one of the relations $B_{(k+1)}, \dots, B_n$ and $A_{(k+1)}, \dots, A_n$ are all entities. Let us suppose further that none of the arguments in the conceptual relation " $B_1 r B_2, \dots, B_k, B_{(k+1)}, \dots, B_n$ " has an indexed quantifier. Then, the conceptual relation " $B_1 r B_2, \dots, B_k, B_{(k+1)}, \dots, B_n$ " implies the conceptual relation " $A_1 r A_2, \dots, A_k, A_{(k+1)}, \dots, A_n$," if A_1, A_2, \dots, A_k and B_1, B_2, \dots, B_k satisfy the conditions of Theorem 1, and for each $i, i = k+1, \dots, n$, the conceptual relation B_i implies the conceptual relation A_i , also according to the criteria of Theorem 1.

Proof

The proof of this theorem follows along the same lines as that of Theorem 1. We only give a brief outline of it here. Under the hypotheses of the theorem, we first show that, for any $j, j = (k+1), \dots, n$, the relation

$$(I) \text{ } "B_1 r B_2, \dots, B_k, B_{(k+1)}, \dots, B_{(j-1)}, (B_{j.1} r B_{j.2}, \dots, B_{j.q}, \dots, B_{j.m}), B_{(j+1)}, \dots, B_n"$$

Implies the relation

$$"B_1 r B_2, \dots, B_k, B_{(k+1)}, \dots, B_{(j-1)}, (B_{j.1} r B_{j.2}, \dots, A_{j.q}, \dots, B_{j.m}), B_{(j+1)}, \dots, B_n"$$

for any $q, q=1, \dots, m$. Since q is any integer between 1 and m , we can apply this argument m times to conclude that the relation (I), above, implies the relation

$$"B_1 r B_2, \dots, B_k, B_{(k+1)}, \dots, B_{(j-1)}, (A_{j.1} r A_{j.2}, \dots, A_{j.q}, \dots, A_{j.m}), B_{(j+1)}, \dots, B_n"$$

Since j is any integer between $(k+1)$ and n , we can, in turn, apply this argument $(n-k)$ times to conclude that the relation (I), above, implies the relation

$$(II) "B_1 r B_2, \dots, B_k, A(k+1), \dots, A(j-1), (A_{j.1} r_j A_{j.2}, \dots, A_{j.q}, \dots, A_{j.m}), A(j+1), \dots, A_n"$$

Finally, k applications of Lemma 1 to relation (II), allows us to conclude that relation (I) implies the relation

$$"A_1 r A_2, \dots, A_k, A(k+1), \dots, A(j-1), (A_{j.1} r_j A_{j.2}, \dots, A_{j.q}, \dots, A_{j.m}), A(j+1), \dots, A_n"$$

which proves the theorem.

In theorems 1 and 2 we imposed the restriction that the quantifiers in the conceptual relations not be indexed quantifiers. We would like to consider now the case in which some of the arguments of a conceptual relation do have an indexed quantifier. As we have seen, these relations correspond to sentences like "all americans want to own a house," "the human body kills germs by producing chemicals," and "all humans like themselves." We show how indexed quantifiers are dealt with in order to draw inferences based on the notion of classification.

Definition 5.1

Two indexed quantifiers are said to be the same, if they are both existential or both universal, and they have the same index (x, y, z , etc.).

Theorem 3

Let

$$(I) B_1 r B_2, \dots, B_k, B(k+1), \dots, B_n$$

and

$$(II) A_1 r A_2, \dots, A_k, A(k+1), \dots, A_n$$

be conceptual relations, where B_1, \dots, B_k , and A_1, \dots, A_k denote entities, $B(k+1), \dots, B_n$ and $A(k+1), \dots, A_n$ denote conceptual relations, r is a relation, and the arguments in each one of the relations $B(k+1), \dots, B_n$ and $A(k+1), \dots, A_n$ are all entities. Let us suppose further that concept B_1 has an indexed quantifier, and let $Q_b = \{B_{u1}, B_{u2}, \dots, B_{ug}, B_{v1.w1}, B_{v2.w2}, \dots, B_{vh.wh}\}$ be the set of arguments in the conceptual relation (I) that have the same indexed quantifier as B_1 (here, the symbols $B_{v.w}$ refer to arguments in the embedded relations $B(k+1), \dots, B_n$). Then, the conceptual relation (I) implies the conceptual relation (II), if conditions (a) and (b), below, are satisfied:

- (a) A1 has an indexed quantifier in relation (II), and the set of arguments in (II) that have the same indexed quantifier as A1 is $Q_a = \{A_{u1}, A_{u2}, \dots, A_{ug}, A_{v1.w1}, A_{v2.w2}, \dots, A_{vh.wh}\}$.
- (b) If (I') and (II') are the relations obtained from (I) and (II), respectively, by eliminating the indices of all indexed quantifiers, then (I') implies (II') according to Theorem 2.

Furthermore, except for the indices, the quantifiers of A_1, A_2, \dots, A_k are determined according to the criteria of Theorem 1.

Proof

Again, we only outline the proof of this theorem. First, we take care of all arguments with indexed quantifiers. We note that the arguments in the set Q_b are all equal to B_1 and, similarly, all arguments in the set Q_a are equal to A_1 . Next, let R be the conceptual relation obtained from (I) by replacing each argument that belongs to the set Q_b with A_1 . Let Q_r be the set of all arguments in R with the same indexed quantifier as A_1 . Then, since B_1 and A_1 satisfy the conditions of Theorem 1, four cases are possible.

(1) $A_1 = B_1$.

In this case, clearly (I) implies R , since they are identical. Furthermore, the quantifiers of the arguments that belong to the set Q_r are all the same, and equal to the indexed quantifier of B_1 .

(2) B_1 is a proper subclass of A_1 .

To show that (I) implies R , let us assume that relation (I) holds true, that is, that it exists in LTM. If (I) holds true and B_1 is a proper subclass of A_1 , then R is true for some entities in the class A_1 , namely those in B_1 . Thus, (I) implies R , and all the arguments in Q_r have the same indexed existential quantifier.

(3) A_1 is a proper subclass of B_1 , and A_1 inherits the relation R from B_1 .

If (I) holds true, because of inheritance (I) must hold true for all entities in the class B_1 and, in particular, for all entities in class A_1 , since A_1 is a proper subclass of B_1 . Thus, (I) implies R , and all arguments of R that belong to the set Q_r have the same universal quantifier.

(4) C1 is a concept which is a proper subclass of both A1 and B1, and C1 inherits from B1 the conceptual relation S obtained by replacing in (I) each argument that belongs to the set Qb with C1. We let Qs be the set of all arguments in S with the same indexed quantifier as C1.

Applying case (3), above, to the conceptual relations (I) and S, we conclude that (I) implies S, and all arguments of S that belong to the set Qs have the same indexed universal quantifier. Applying case (2) to the conceptual relations S and R, we conclude that S implies R, and all arguments of R that belong to the set Qr have the same indexed existential quantifier. Thus, (I) implies R, where all the arguments of R that belong to the set Qr have the same indexed existential quantifier.

This shows that under the hypotheses of the theorem, conceptual relation (I) implies conceptual relation R. All that remains now is to note that R and (II) may only differ in arguments whose quantifiers are not indexed, so Theorem 2 can be applied to conclude that R implies (II), and therefore that (I) implies (II).

Examples

Let us suppose that the following *action-structure* exists in LTM, and consider the two examples below.

```
(a1
  (args (human)(human))
  (pr (like))
  (actor (human (q1 (allx))))
  (object (human (q2 (allx)))))
```

This structure represents the fact that "everyone likes himself/herself." In the notation of Theorem 3, B1=human, r=like, B2=human, B1 has an indexed universal quantifier, and Qb = { B1, B2 }.

a) Suppose the question "does everyone like everyone?" is asked. This question is represented by the structure

```
(a2
  (args (human)(human))
  (pr (like))
  (actor (human (q (all))))
  (object (human (q (all)))))
```

and in the notation of Theorem 3, A1=human, r=like, A2=human, and the quantifiers of A1 and A2 are universal, but not indexed. Therefore, the relations "A1 r A2" and "B1 r B2" do not satisfy con-

dition (a) of Theorem 3, and the question can not be answered affirmatively.

b) Apart from structure a1, above, let us suppose that the relation underlying the sentence "all americans are humans" also exists in LTM. Consider the question "do all americans like themselves?" This question is represented by the structure

```
(a3
  (args (american)(american))
  (pr (like))
  (actor (american (q (allx))))
  (object (american (q (allx)))))
```

In the notation of Theorem 3, A1=american, r=like, A2=american, A1 has an indexed quantifier, and $Qa = \{ A1, A2 \}$. Thus, condition (a) of Theorem 3 is satisfied. Since "american" is a proper subclass of "human," and it inherits the relation underlying "all americans like themselves" from "human," condition (b) of the theorem is also satisfied, and the answer to the question is affirmative.

Table 2 contains the algorithm for answering verification questions using the inference criteria we have explained in this section. Let us illustrate the main ideas in the algorithm with an example. Suppose the question asked is "do germs cause diseases?" First, we look in the *object-structure* of *germ*, and ask the question "what do germs cause?" If *disease* is one of the things caused by germs, then the answer will be yes. If not, let *c* be something caused by *germs* (as indicated in the *object-structure* of *germ*). If *c* is a *disease* then the answer to the question is yes. If not, we determine if *disease* is a descendant of *c*, in which case, if inheritance can be verified, then the answer will be yes. If this also fails, we determine if there is a common descendant of *disease* and *c* which inherits the desired conceptual relation from *c*, in which case the answer is yes. If all these fail, we examine the ancestors of *germ*. We ask, "is there an ancestor of the concept *germ* which causes diseases and such that *germ* inherits this conceptual relation from it?" If so, the answer will be yes. If this also fails, we ask "is there anything which is a germ and which causes diseases?" If so, again the answer will be yes. Finally, if this also fails, we try to find out if something which is a *germ* inherits the conceptual relation "cause diseases" from an ancestor other than *germs*.

Consider the verification question represented by the conceptual relation A given by " A_1 r A_2, \dots, A_k ." For any conceptual relation, say S , let Q_s be either the empty set, if the quantifier of the first argument of S is not indexed, or the set of arguments in S with the same indexed quantifier as the first argument of S .

Initialization Set the list VISITED to nil.

Step 1 Let W be the set of all conceptual relations of the form " A_1 r B_2, \dots, B_m " that are present in the *object-structure* of A_1 in LTM.

For each conceptual relation w in W , for which $Q_w = Q_a$

For each j in $\{2, \dots, k\}$

If A_j and B_j are not conceptual relations, then

If $A_j = B_j$, or B_j is a subclass of A_j , or A_j is a subclass of B_j and A_j inherits the conceptual relation " A_1 r $B_2, \dots, B_j, \dots, B_k$ " from B_j , or the children of B_j inherit the conceptual relation " A_1 r $B_2, \dots, B_j, \dots, B_k$ " from B_j , and A_j and B_j have a common descendant, then continue with the next j in $\{2, \dots, k\}$;

else if A_j and B_j are conceptual relations and B_j implies A_j , then

continue with the next j in $\{2, \dots, k\}$;

else continue with the next w in W ;

Answer "yes"; exit;

Add A_1 to VISITED.

Step 2 Let PARENTS be the set of concepts p such that the conceptual relation " A_1 is-a p " is present in LTM.

For each p in PARENTS

If p is the root of the hierarchy or a member of VISITED, continue with the next p in PARENTS.

Let W be the set of all conceptual relations of the form " p r B_2, \dots, B_m " that are present in the *object-structure* of p in LTM.

For each w in W for which A_1 inherits the conceptual relation " A_1 r B_2, \dots, B_m " from p , and $Q_w = Q_a$,

For each j in $\{2, \dots, k\}$

If A_j and B_j are not conceptual relations, then

If $A_j = B_j$, or B_j is a subclass of A_j , or A_j is a subclass of B_j and A_j inherits the conceptual relation " p r $B_2, \dots, B_j, \dots, B_k$ " from B_j , or the children of B_j inherit the conceptual relation " p r $B_2, \dots, B_j, \dots, B_k$ " from B_j , and A_j and B_j have a common descendant, then continue with the next j in $\{2, \dots, k\}$;

else if A_j and B_j are conceptual relations and B_j implies A_j , then

continue with the next j in $\{2, \dots, k\}$;

else continue with the next w in W ;

Answer "yes"; exit;

Add p to VISITED;

Append the parents of p to PARENTS;

Step 3 Let CHILDREN be the set of concepts c such that the conceptual relation " A_1 classes-of c " is present in LTM.

For each c in CHILDREN

Perform Step 1 with c in place of A_1 ;

Perform Step 2 with c in place of A_1 ;

Append the children of c to CHILDREN;

Step 4 Answer "I don't know"; exit.

Table 2. Algorithm for Inferring the Answer from the Classification Links in LTM

6. Inferring the Answer by Classifying the Concepts in the Question

In this section, we discuss the inferences drawn when a question is asked about a concept not existing in LTM. Suppose that a question is asked about a concept, say *C*, that is not present in LTM. Normally, nothing can be said about such a concept. However, if the question itself contains a description of the concept, then such a description may be used by the *Classifier* to determine which concepts in LTM would be the parents and which would be the children of concept *C*, if *C* were present in LTM. Knowing what the parents and children of a concept are allows the kinds of inferences discussed in the previous section, even if the concept itself does not have an entry in LTM. We illustrate this situation with an example. Let us suppose that LTM contains the representation of the concepts and conceptual relations underlying the sentences below. This representation is shown in Figure 7.a.

All birds are animals. All animals which live in the Antarctic swim.

Suppose now we ask the question "Do birds which live in the Antarctic swim?" The question concerns the concept *birds which live in the Antarctic*, which does not exist in LTM. However, because the concept is described by the question in terms of other concepts which do exist in LTM (*birds* and *Antarctic*), an attempt can be made to classify the concept in the question with respect to the concepts in LTM. The *Classifier* and *Integration* algorithms are discussed in detail in [10]. We now describe briefly the aspects of the *Classifier* that are relevant to the inference

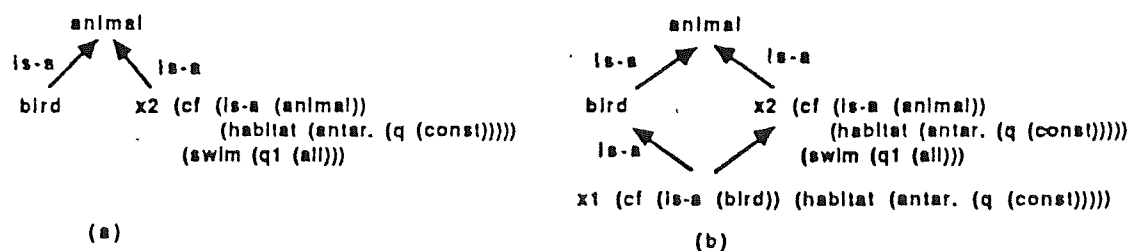


Figure 7. Representation of "All birds are animals. All animals which live in the Antarctic swim." (a) before asking the question "Do birds which live in the Antarctic swim?" (b) after asking the question

mechanisms.

As shown in Section 2, the concept *birds which live in the Antarctic* is represented by an *object-structure* that has a *characteristic-features* slot, as follows:

(x1 (cf (is-a (bird)) (habitat (Antarctic (q (constant))))))

In order to determine what the parents and children of a concept are, the *Classifier* must be able to compare two concepts and determine whether they are the same, whether one is a subclass of the other, or whether there is no hierarchical relation between them. This comparison is based on the following observations: a) if x is a concept whose representation structure contains a *characteristic-features* slot, then a concept y represents a subclass of concept x if each characteristic feature of x is true of all entities in the class corresponding to concept y ; b) if x is a concept whose representation structure does not contain a *characteristic-features* slot, then a concept y represents a subclass of concept x if the conceptual relation " y is-a x " holds true, or if there exists a sequence of concepts y_1, y_2, \dots, y_n , such that " y is-a y_1 ," " y_1 is-a y_2 ," ..., " $y_{(n-1)}$ is-a y_n ," and " y_n is-a x " all hold true.

Given a concept like x_1 , above, whose location in LTM is to be determined, the *Classifier* applies the above criteria to find the parents and children of x_1 . However, the *Classifier* must do this without having to compare x_1 with all or most concepts in LTM. The search for the parents and children of x_1 is first restricted by the contents of the *is-a* slot in the representation structure of x_1 . Since this *is-a* slot contains the concept *bird*, then at least one parent of x_1 must be either the concept *bird* or a descendant of *bird*, and every child of x_1 must be a descendant of *bird*. Thus, the *Classifier* first compares concept x_1 with a relatively small part of LTM, namely, the sub-hierarchy consisting of concept *bird* and its descendants. It is possible though, as our example illustrates, that concept x_1 be related to other concepts in LTM which are not part of the *bird* sub-hierarchy. In general, this might be the largest part of LTM so it is essential to have a way of reducing this search to a minimum number of concepts in LTM. To this objective, we observe the following: a) since all children of x_1 must also be descendants of *bird*, (that is, they must be part of the *bird* sub-hierarchy, outside the *bird* sub-hierarchy we can only find parents of x_1 ; b) from the representation of x_1 we see that if a concept y is a parent of x_1 , then y must satisfy the conceptual relation

"y habitat *Antarctic*." Therefore, the *Classifier* only needs to examine those concepts in LTM that satisfy this conceptual relation. How can these concepts be found? As we recall from Section 2, every conceptual relation is linked to all of its arguments. Therefore, if the conceptual relation "y habitat *Antarctic*" holds true for some concept y, then the *object-structure* of *Antarctic* in LTM must contain the conceptual relation "*Antarctic* habitat-of y," and finding the concepts outside the *bird* sub-hierarchy that might be related to x1 reduces to examining the *object-structure* of *Antarctic*. When this fails, the ancestor concepts of *Antarctic* are also examined. In our example, the *Classifier* determines that concept x1 is a child of *bird* and also a child of x2 (*animals which live in the Antarctic*), as shown in Figure 7.b. With this knowledge, it can now be inferred that "birds which live in the Antarctic swim," because "birds which live in the Antarctic are animals which live in the Antarctic" and "all animals which live in the Antarctic swim."

As a second example, suppose the concepts and conceptual relations underlying the following sentences have been stored in LTM:

All mammals are animals. All whales are mammals. All whales live in the sea. Some whales feed on fish.

The representation of these concepts is shown in Figure 8.a. Suppose we now ask the question "do animals which live in the sea feed on fish?" Because the question concerns the concept *animals which live in the sea*, which does not exist in LTM, the *Classifier* is activated, which determines that concept *animals which live in the sea* is a child of *animal* and a parent of *whale*, as illustrated in Figure 8.b. With this knowledge, the question can now be answered as follows: "Yes, some animals which live in the sea feed on fish because whales are animals which live in the sea and some whales feed on fish." Similarly, if the conceptual relations:

All infections are diseases. The human-body recognizes germs which cause infections.

are present in LTM then the question "Does the human-body recognize germs which cause diseases?" can be answered by invoking the *Classifier* (which will determine that the concept *germs which cause diseases* is a superclass of *germs which cause infections*), through the following reasoning: "Yes, the human-body recognizes some germs which cause diseases because the human-body recognizes germs which cause infections, and all infections are diseases." These

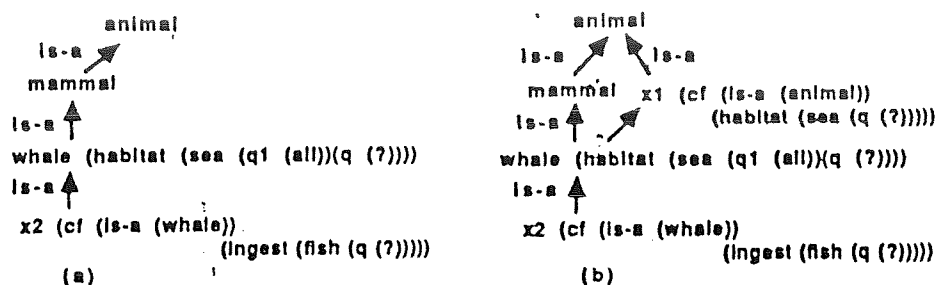


Figure 8. Representation of "All mammals are animals. All whales are mammals. All whales live in the sea. Some whales feed on fish." (a) before asking the question "Do animals which live in the sea feed on fish?" (b) after asking the question

examples show how the notion of classification is used as the basis of an inference mechanism when the question involves a concept not present in LTM.

7. Types of Questions Handled by SNOWY

Lehnert [15] identified the following question categories: causal antecedent, goal orientation, causal consequent, verification, instrumental, concept completion, expectational, feature specification, quantification, enablement, disjunction, judgmental and request. Kolodner [14] added time, setting, identification and duration. Dyer [5] included event specification and empathy. We have adopted this categorization with very minor changes. We have added to these categories the following types: definition ("what are bacteria?") and classification ("are bacteria germs?"), a subclass of verification. Also, verification questions of the type exemplified by "do all antibodies kill germs" are further classified as quantificational questions. The present implementation of SNOWY handles the following question categories: verification, classification, definition, feature specification, quantificational, goal, location, instrumental, identification, and concept completion.

The structures below are built for the questions "do all germs cause infections?," "why does the human body produce antibodies?," "does the human body kill germs by producing antibodies?," and "do all americans want to own a house?," respectively.

```

(Verify (germ cause Infection
  (a1 (args (germ)(Infection))(pr (cause))
    (actor (germ (q (all))))(object (Infection (q (?)))))))

(Purpose (human-body make antibody
  (a1 (args (human-body)(antibody))(pr (make))
    (actor (human-body (q (?))))(object (antibody (q (?))))))

(Verify (human-body cause-to-die germ
  (a1 (args (human-body)(germ))(pr (cause-to-die))
    (actor (human-body (q (?x))))(object (germ (q (?))))
    (instrument (e1)))
  (e1 (args (human-body)(antibody))(pr (make))
    (actor (human-body (q (?x)))) (object (antibody (q (?))))))

(Verify (american want e1
  (a1 (args (american)(e1)) (pr (want))
    (actor (american (q (allx)))) (object (e1 (q (some))))))
  (e1 (args (american)(house)) (pr (own))
    (actor (american (q (allx)))) (object (house (q (some))))))

```

Verification questions fall under the schema *Verify A1 r1 A2*, where A1 and A2 stand for concepts and r1 for relation. Answers for these questions are obtained by searching for the relation *r1 A2* under the concept A1. If a quantifier or a case is part of the question (like in the examples above), those slots in the *action-structure* are examined. For instance, a quantificational question will be answered by searching the *object-structure* for that concept and then by examining the content of the quantifier slots in the *action-structure*. Purpose questions fall under the schema *Purpose A1 r1 A2*. These questions are answered by accessing the "purpose" slot in the *action-structure* representation of the conceptual relation *A1 r1 A2*.

Since conceptual relations are linked to all its arguments, answering identification questions becomes a very straightforward procedure. Thus, for a question like "what causes diseases?" we note that if the conceptual relation "X causes diseases" is true for some concept "X," then each such relation will also be indexed under the concept "diseases," so when trying to answer this question, we look in the LTM entry for "diseases" under the relation "cause:by." Anything found in this slot will be an answer to the question. Likewise, consider the sentence "the human body destroys germs by producing antibodies." The conceptual relation described by this sentence is linked to "human body," to "germs," and to "antibodies," so it can be accessed through any of these concepts. If this were not so, questions like "what do you know about antibodies?" or "for what purpose does the human body produce antibodies?" could not be answered.

8. Discussion

In summary, we have shown a method to represent and quantify in relations with multiple arguments, some of which can also be relations. We have also indicated how necessary, necessary and sufficient, and contingent information can be associated with a given concept. We have provided FOPC formulae for each of the knowledge structures presented in this paper. Finally, we have proven that the algorithm draws valid inferences from the representation structures.

The representation language which has been described here is a subset of FOPC. However, the interesting aspect of this language lies in the reasoning algorithms which it permits and in its representation capabilities. It is also relevant to note that one of the main motivations behind the design of this language has been the acquisition of knowledge from texts, which involves the construction of knowledge representation structures from scratch. And this in turn requires recognition and integration algorithms which will be very hard to accommodate into a knowledge representation scheme based exclusively on first order logic.

One intriguing aspect of these ideas is that deep semantic relations such as classification hierarchies underly syntactical constructions like restrictive relative clauses. The relevance of explicit classification has been widely observed by cognitive scientists working on comprehension. In explicit classification, the classes are clearly introduced and named, as in the passage:

There are two kinds of mammals: terrestrial mammals and aquatic mammals. Aquatic mammals include the carnivorous and the omnivorous.

The classification underlying restrictive relative clauses is a more basic cognitive phenomenon than explicit classification. In this paper, we have shown that it is a necessary element in deriving certain kinds of inferences. In [11], we have shown that this type of classification is also an essential component for the acquisition of knowledge from texts.

One may wonder if concepts underlying relative clauses and other restrictive qualifiers should have the same status in memory as concepts for which "we have words." We assume that "we have words" means single words, since we have indicated that complex noun groups are also represented as classification hierarchies. The answer is an unqualified "yes." The concept *car* is expressed currently by a single word, namely, "car." The concepts expressed by the phrases "cars

with carburetors" and "cars with fuel injection" are as natural as the concept *car*. If these concepts are not classified under the concept *car*, many inferences could not be accomplished and, as a consequence, understanding will not take place.

From a methodological point of view, this paper is not a logicist paper, since the program which embodies the theory does not use logic to derive inferences. In fact, a main emphasis in this research has been to discover alternative methods to logic, which integrate indexing and inference mechanisms. We have shown that certain deductive problems can be solved by the right representation of knowledge and by keeping memory organized in a principled manner. The boundaries of logic are not the boundaries of rationality. However, we do think that logic is an excellent tool to convey the meaning and scope of theories.

One of the main applications of these ideas, which also constitutes a natural theoretical extension, is to the task of knowledge acquisition from texts for expert systems. We are presently building a knowledge acquisition system which builds automatically from texts certain kinds of expert systems. The domain expert is instructed to describe its subject in a top down manner. The system builds automatically the classification hierarchies from the text and, then, the reasoning techniques explained in this paper are used to answer questions posed by users of the expert system. In this system, comprehension, memory reorganization and problem-solving are integrated processes, since all of them share the same algorithms. (See [11] for a discussion of these ideas.)

Acknowledgement

This paper has benefited from the insightful and critical comments of an anonymous referee. He/she is gratefully acknowledged.

References

- [1] J. Allen and C. R. Perrault, "Analyzing Intention in Utterances," *Artificial Intelligence*, vol. 15 (1), pp. 143-178, 1980.
- [2] R. Brachman and J. Schmolze, "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science*, vol. 9, pp. 171-216, 1985.
- [3] R. Carnap. *Meaning and Necessity*. University of Chicago Press, 1947.
- [4] E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*. Reading: Addison-Wesley, 1985.
- [5] M. Dyer, *In-Depth Understanding*. Cambridge: MIT Press, 1983.
- [6] F. Gomez, "Towards a Theory of Comprehension of Declarative Texts," In *Proceedings of the 20th Meeting of the Association of Computational Linguistics*, Toronto: ACL, 1982, pp. 36-43.
- [7] F. Gomez, "A Model of Comprehension of Elementary Scientific Texts," In *Proceedings of the Workshop on Theoretical Approaches to Natural Language Understanding*, Halifax, Nova Scotia, 1985, pp. 70-81.
- [8] F. Gomez, "WUP: A Parser Based in Word Usage," In *Proceedings Phoenix Conference on Computers and Communication*, Scottsdale: IEEE, 1988, pp. 445-449.
- [9] F. Gomez and C. Segami, "SNOWY: A System Which Understands Elementary Scientific Texts," Technical Report CS-TR-87-04, Department of Computer Science, University of Central Florida, Orlando, Florida, 1987.
- [10] F. Gomez and C. Segami, "The Recognition and Classification of Concepts In Understanding Scientific Texts," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 1, pp. 51-77, 1989.
- [11] F. Gomez, "Knowledge Acquisition From Natural Language For Expert Systems Based On Classification Problem-Solving Methods," Fourth AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop," Banff, Canada, 1989.

- [12] B. J. Grosz, A.K. Joshi, and S. Weinstein, "Providing an Unified Account of Definite Noun Phrases In Discourse," In Proceedings, ACL, 1983, pp. 44-50.
- [13] B. Grosz and C. Sidner, "Attention, Intention, and the Structure of Discourse," *Computational Linguistics*, Vol. 12, 1986.
- [14] J. Kolodner, "*Retrieval and Organizational Strategies in Conceptual Memory*," Technical Report 187. Dept. of Computer Science, Yale University, 1980.
- [15] W. Lehnert, *The Process of Question Answering*. Hillsdale: Lawrence Erlbaum, 1978.
- [16] Litman, D.J. and J.F. Allen. "A Plan Recognition Model for Subdialogues In Conversations," *Cognitive Science*, vol. 11,2, pp. 163-200, 1987.
- [17] W. V. O. Quine, "On what there is," in *From a logical point of view*. Cambridge: Harvard University Press, 1964.
- [18] R. Schank, *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1982.
- [19] J. Schmolze and T. Lipkis, "Classification In the KL-ONE Knowledge Representation System," In *Proceedings of the Elghth International Joint Conference on Artificial Intelligence*, Karlsruhe: IJCAI, 1983, pp. 330-332.
- [20] B. L. Webber, "The Interpretation of Tense In Discourse," In Proceedings, ACL, 1987, pp. 147-154.
- [21] R. Wilensky, *Planning and Understanding*. Reading: Addison-Wesley, 1983.